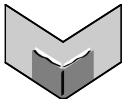
The Journal of Machine Learning Research
Volume 11
Print-Archive Edition

Pages 1833–3680



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 11
Print-Archive Edition

The Journal of Machine Learning Research (JMLR) is an open access journal. All articles published in JMLR are freely available via electronic distribution. This Print-Archive Edition is published annually as a means of archiving the contents of the journal in perpetuity. The contents of this volume are articles published electronically in JMLR in 2010.

JMLR is abstracted in ACM Computing Reviews, INSPEC, and Psychological Abstracts/PsycINFO.

JMLR is a publication of Journal of Machine Learning Research, Inc. For further information regarding JMLR, including open access to articles, visit http://www.jmlr.org/.

JMLR Print-Archive Edition is a publication of Microtome Publishing under agreement with Journal of Machine Learning Research, Inc. For further information regarding the Print-Archive Edition, including subscription and distribution information and background on open-access print archiving, visit Microtome Publishing at http://www.mtome.com/.

ISSN 1532-4435 (print)
ISSN 1533-7928 (online)

JMLR Editorial Board

Journal of Machine Learning Research

Volume 11, 2010

Journal of Machine Learning Research 11 (2010) 1833-1863 Submitted 10/09; Revised 5/10; Published 6/10

Permutation Tests for Studying Classifier Performance

Markus Ojala MARKUS.OJALA@TKK.FI
Helsinki Institute for Information Technology
Department of Information and Computer Science
Aalto University School of Science and Technology
P.O. Box 15400, FI-00076 Aalto, Finland

Gemma C. Garriga GEMMA.GARRIGA@LIP6.FR
Université Pierre et Marie Curie
Laboratoire d'Informatique de Paris 6
4 place Jussieu, 75005 Paris, France

Editor: Xiaotong Shen

Abstract

We explore the framework of permutation-based p-values for assessing the performance of classifiers. In this paper we study two simple permutation tests. The first test assess whether the classifier has found a real class structure in the data; the corresponding null distribution is estimated by permuting the labels in the data. This test has been used extensively in classification problems in computational biology. The second test studies whether the classifier is exploiting the dependency between the features in classification; the corresponding null distribution is estimated by permuting the features within classes, inspired by restricted randomization techniques traditionally used in statistics. This new test can serve to identify descriptive features which can be valuable information in improving the classifier performance. We study the properties of these tests and present an extensive empirical evaluation on real and synthetic data. Our analysis shows that studying the classifier performance via permutation tests is effective. In particular, the restricted permutation test clearly reveals whether the classifier exploits the interdependency between the features in the data.

Keywords: classification, labeled data, permutation tests, restricted randomization, significance testing

1. Introduction

Building effective classification systems is a central task in data mining and machine learning. Usually, a classification algorithm builds a model from a given set of data records in which the labels are known, and later, the learned model is used to assign labels to new data points. Applications of such classification setting abound in many fields, for instance, in text categorization, fraud detection, optical character recognition, or medical diagnosis, to cite some.

For all these applications, a desired property of a good classifier is the power of generalization to new, unknown instances. The detection and characterization of statistically significant predictive patterns is crucial for obtaining a good classification accuracy that generalizes beyond the training data. Unfortunately, it is very often the case that the number of available data points with labels is not sufficient. Data from medical or biological applications, for example, are characterized by high

o	x	x	x	x	x	x	x	+
x	x	o	x	x	x	x	o	+
x	x	x	x	o	o	x	x	+
x	x	x	x	x	x	x	o	+
x	x	o	x	o	o	o	x	+
x	x	x	x	x	x	x	o	+
x	o	o	x	o	x	x	x	+
x	x	x	x	o	x	x	o	+
o	o	o	x	x	o	o	o	−
o	o	o	o	o	o	o	o	−
x	o	x	o	o	o	o	o	−
x	o	x	o	o	x	o	o	−
o	o	x	o	o	o	o	o	−
o	o	o	o	o	o	x	o	−
x	o	o	o	o	o	o	o	−
o	o	o	x	o	o	o	o	−

Data Set D_1

x	x	x	o	x	x	x	x	+
x	x	x	x	o	x	x	x	+
x	x	x	x	x	x	x	x	+
x	o	x	x	x	x	x	x	+
o	o	o	o	o	o	o	x	+
x	o	o	o	o	o	o	o	+
o	o	o	o	o	x	o	o	+
o	o	o	o	o	o	o	o	+
x	x	x	x	o	o	o	x	−
x	x	x	x	x	o	o	o	−
x	x	o	x	o	o	o	o	−
x	x	x	x	o	o	o	o	−
o	o	o	o	x	x	x	x	−
o	o	o	o	x	x	x	x	−
o	x	o	o	x	x	x	o	−
o	o	o	x	x	x	x	x	−

Data Set D_2

Figure 1: Examples of two 16×8 nominal data sets D_1 and D_2 each having two classes. The last column in both data sets denotes the class labels $(+, -)$ of the samples in the rows.

dimensionality (thousands of features) and small number of data points (tens of rows). An important question is whether we should believe in the classification accuracy obtained by such classifiers.

The most traditional approach to this problem is to estimate the error of the classifier by means of cross-validation or leave-one-out cross-validation, among others. This estimate, together with a variance-based bound, provides an interval for the expected error of the classifier. The error estimate itself is the best statistics when different classifiers are compared against each other (Hsing et al., 2003). However, it has been argued that evaluating a single classifier with an error measurement is ineffective for small amount of data samples (Braga-Neto and Dougherty, 2004; Golland et al., 2005; Isaksson et al., 2008). Also classical generalization bounds are not directly appropriate when the dimensionality of the data is too high; for these reasons, some recent approaches using filtering and regularization alleviate this problem (Rossi and Villa, 2006; Berlinet et al., 2008). Indeed, for many other general cases, it is useful to have other statistics associated to the error in order to understand better the behavior of the classifier. For example, even if a classification algorithm produces a classifier with low error, the data itself may have no structure. Thus the question is, how can we trust that the classifier has learned a significant predictive pattern in the data and that the chosen classifier is appropriate for the specific classification task?

For instance, consider the small toy example in Figure 1. There are two nominal data matrices D_1 and D_2 of sizes 16×8. Each row (data point) has two different values present, x and o. Both data sets have a clear separation into the two given classes, + and −. However, it seems at first sight that the structure within the classes for data set D_1 is much simpler than for data set D_2. If we train a 1-Nearest Neighbor classifier on the data sets of Figure 1, we have that the classification error (leave-one-out cross-validation) is 0.00 on both D_1 and D_2. However, is it true that the classifier is using a real dependency in the data? Or are the dependencies in D_1 or D_2 just a random artifact of

some simple structure? It turns out that the good classification result in D_1 is explained purely by the different value distributions inside the classes whereas in D_2 the interdependency between the features is important in classification. This example will be analyzed in detail later on in Section 3.3.

In recent years, a number of papers have suggested to use permutation-based p-values for assessing the competence of a classifier (Golland and Fischl, 2003; Golland et al., 2005; Hsing et al., 2003; Jensen, 1992; Molinaro et al., 2005). Essentially, the permutation test procedure measures how likely the observed accuracy would be obtained by chance. A p-value represents the fraction of random data sets under a certain null hypothesis where the classifier behaved as well as or better than in the original data.

Traditional permutation tests suggested in the recent literature study the null hypothesis that the features and the labels are independent, that is, that there is no difference between the classes. The null distribution under this null hypothesis is estimated by permuting the labels of the data set. This corresponds also to the most traditional statistical methods (Good, 2000), where the results on a control group are compared against the results on a treatment group. This simple test has been proven effective already for selecting relevant genes in small data samples (Maglietta et al., 2007) or for attribute selection in decision trees (Frank, 2000; Frank and Witten, 1998). However, the related literature has not performed extensive experimental studies for this traditional test in more general cases.

The goal of this paper is to study permutation tests for assessing the properties and performance of the classifiers. We first study the traditional permutation test for testing whether the classifier has found a real class structure, that is, a real connection between the data and the class labels. Our experimental studies suggest that this traditional null hypothesis leads to very low p-values, thus rendering the classifier significant most of the time even if the class structure is weak.

We then propose a test for studying whether the classifier is exploiting dependency between some features for improving the classification accuracy. This second test is inspired by restricted randomization techniques traditionally used in statistics (Good, 2000). We study its relation to the traditional method both analytically and empirically. This new test can serve as a method for obtaining descriptive properties for classifiers, namely whether the classifier is using the feature dependency in the classification or not. For example, many existing classification algorithms are like black boxes whose functionality is hard to interpret directly. In such cases, indirect methods are needed to get descriptive information for the obtained class structure in the data.

If the studied data set is known to contain useful feature dependencies that increase the class separation, this new test can be used to evaluate the classifier against this knowledge. For example, often the data is gathered by a domain expert having deeper knowledge of the inner structure of the data. If the classifier is not using a known useful dependency, the classifier performance could be improved. For example, with medical data, if we are predicting the blood pressure of a person based on the height and the weight of the individual, the dependency between these two features is important in the classification as large body mass index is known to be connected with high blood pressure. However, both weight and height convey information about the blood pressure but the dependency between them is the most important factor in describing the blood pressure. Of course, in this case we could introduce a new feature, the body mass index, but in general, this may not be practical; for example, introducing too many new features can make the classification ineffective or too time consuming.

If nothing is known previously from the structure of the data, Test 2 can give some descriptive information for the obtained class structure. This information can be useful as such for understanding

the properties of the classifier, or it can guide the search towards an optimal classifier. For example, if the classifier is not exploiting the feature dependency, there might be no reason to use the chosen classifier as either more complex classifiers (if the data contains useful feature dependencies) or simpler classifiers (if the data does not contain useful feature dependencies) could perform better. Note, however, that not all feature dependencies are useful in predicting the class labels. Therefore, in the same way that traditional permutation tests have already been proven useful for selecting relevant features in some contexts as mentioned above (Maglietta et al., 2007; Frank, 2000; Frank and Witten, 1998), the new test can serve for selecting combinations of relevant features to boost the classifier performance for specific applications.

The idea is to provide users with practical p-values for the analysis of the classifier. The permutation tests provide useful statistics about the underlying reasons for the obtained classification result. Indeed, no test is better than the other, but all provide us with information about the classifier performance. Each p-value is a statistic about the classifier performance; each p-value depends on the original data (whether it contains some real structure or not) and the classifier (whether it is able to use certain structure in the data or not).

The remaining of the paper is organized as follows. In Section 2, we give the background to classifiers and permutation-test p-values, and discuss connections with previous related work. In Section 3, we describe two simple permutation methods and study their behavior on the small toy example in Figure 1. In Section 4, we analyze in detail the properties of the different permutations and the effect of the tests for synthetic data on four different classifiers. In Section 5, we give experimental results on various real data sets. Finally, Section 6 concludes the paper.[1]

2. Background

Let X be an $n \times m$ data matrix. For example, in gene expression analysis the values of the matrix X are numerical expression measurements, each row is a tissue sample and each column represents a gene. We denote the i-th row vector of X by X_i and the j-th column vector of X by X^j. Rows are also called observations or data points, while columns are also called attributes or features. Observe that we do not restrict the data domain of X and therefore the scale of its attributes can be categorical or numerical.

Associated to the data points X_i we have a class label y_i. We assume a finite set of known class labels $\mathcal{Y}$, so $y_i \in \mathcal{Y}$. Let D be the set of labeled data $D = \{(X_i, y_i)\}_{i=1}^n$. For the gene expression example above, the class labels associated to each tissue sample could be, for example, "sick" or "healthy".

In a traditional classification task the aim is to predict the label of new data points by training a classifier from D. The function learned by the classification algorithm is denoted by $f : \mathcal{X} \to \mathcal{Y}$. A test statistic is typically computed to evaluate the classifier performance: this can be either the training error, cross-validation error or jackknife estimate, among others. Here we give as an example the leave-one-out cross-validation error,

$$e(f, D) = \frac{1}{n} \sum_{i=1}^{n} \mathrm{I}(f_{D \setminus D_i}(X_i) \neq y_i) \tag{1}$$

1. A shorter version of this paper appears in the proceedings of the IEEE International Conference on Data Mining (Ojala and Garriga, 2009). This is an improved version based on valuable comments by reviewers which includes: detailed discussions and examples, extended theoretical analysis of the tests including statistical power in special case scenarios, related work comparisons and a thorough experimental evaluation with large data sets.

where $f_{D\setminus D_i}$ is the function learned by the classification algorithm by removing the i-th observation from the data and $\mathrm{I}(\cdot)$ is the indicator function.

It has been recently argued that evaluating the classifier with an error measurement is ineffective for small amount of data samples (Braga-Neto and Dougherty, 2004; Golland et al., 2005; Hsing et al., 2003; Isaksson et al., 2008). Also classical generalization bounds are inappropriate when the dimensionality of the data is too high. Indeed, for many other general cases, it is useful to have other statistics associated to the error $e(f,D)$ in order to understand better the behavior of the classifier. For example, even if a consistent algorithm produces a classifier with low error, the data itself may have no structure.

Recently, a number of papers have suggested to use permutation-based p-values for assessing the competence of a classifier. Essentially, the permutation test procedure is used to obtain a p-value statistic from a null distribution of data samples, as described in Definition 1. In Section 3.1 we will introduce two different null hypotheses for the data.

Definition 1 (Permutation-based p-value) *Let $\widehat{D}$ be a set of k randomized versions D' of the original data D sampled from a given null distribution. The empirical p-value for the classifier f is calculated as follows (Good, 2000),*[2]

$$p = \frac{|\{D' \in \widehat{D} : e(f,D') \leq e(f,D)\}| + 1}{k+1}.$$

The empirical p-value of Definition 1 represents the fraction of randomized samples where the classifier behaved better in the random data than in the original data. Intuitively, it measures how likely the observed accuracy would be obtained by chance, only because the classifier identified in the training phase a pattern that happened to be random. Therefore, if the p-value is small enough—usually under a certain threshold, for example, $\alpha = 0.05$—we can say that the value of the error in the original data is indeed significantly small and in consequence, that the classifier is significant under the given null hypothesis, that is, the null hypothesis is rejected.

Ideally the entire set of randomizations of D should be used to calculate the corresponding permutation-based p-value. This is known as the *exact randomization test*; unfortunately, this is computationally infeasible in data that goes beyond toy examples. Instead, we will sample from the set of all permutations to approximate this p-value. It is known that the Monte Carlo approximation of the p-value has a standard deviation of $\sqrt{\frac{p(1-p)}{k}}$, see, for example, Efron and Tibshirani (1993) and Good (2000), where p is the underlying true p-value and k is the number of samples used. Since p is unknown in practice, the upper bound $\frac{1}{2\sqrt{k}}$ is typically used to determine the number of samples required to achieve the desired precision of the test, or the value of the standard deviation in the critical point of $p = \alpha$ where α is the significance level. Alternatively, a sequential probability ratio test can be used (Besag and Clifford, 1991; Wald, 1945; Fay et al., 2007), where we sample randomizations of D until it is possible to accept or reject the null hypothesis. With these tests, often already 30 samples are enough for statistical inference with significance level $\alpha = 0.05$.

2. Notice the addition of 1 in both the denominator and the numerator of the definition. This adjustment is an standard procedure to compute empirical p-values and it is justified by the fact that the original database D is as well a randomized version of itself.

We will specify with more details in the next section how the randomized versions of the original data D are obtained. Indeed, this is an important question as each randomization method entails a certain null distribution, that is, which properties of the original data are preserved in the randomization test, directly affecting the distribution of the error $e(f, D')$. In the following, we will assume that the number of samples k is determined by any of the standard procedures just described here.

2.1 Related Work

As mentioned in the introduction, using permutation tests for assessing the accuracy of a classifier is not new, see, for example, Golland and Fischl (2003), Golland et al. (2005), Hsing et al. (2003) and Molinaro et al. (2005). The null distribution in those works is estimated by permuting labels from the data. This corresponds also to the most traditional statistical methods (Good, 2000), where the results on a control group are compared against the results on a treatment group. This traditional null hypothesis is typically used to evaluate one single classifier at a time (that is, one single model) and we will call it as Test 1 in the next section where the permutation tests are presented.

This simple traditional test has already been proven effective for selecting relevant genes in small data samples (Maglietta et al., 2007) or for attribute selection in decision trees (Frank, 2000; Frank and Witten, 1998). Particularly, the contributions by Frank and Witten (1998) show that permuting the labels is useful for testing the significance of attributes at the leaves of the decision trees, since samples tend to be small. Actually, when discriminating attributes for a decision tree, this test is preferable to a test that assumes the chi-squared distribution.

In the context of building effective induction systems based on rules, permutation tests have been extensively used by Jensen (1992). The idea is to construct a classifier (in the form of a decision tree or a rule system) by searching in the space of several models generated in an iterative fashion. The current model is tested against other competitors that are obtained by local changes (such as adding or removing conditions in the current rules). This allows to find final classifiers with less over-fitting problems. The evaluation of the different models in this local search strategy is done via permutation tests, using the framework of multiple hypothesis testing (Benjamini and Hochberg, 1995; Holm, 1979). The first test used corresponds to permuting labels—that is, Test 1—while the second test is a conditional randomization test. Conditionally randomization tests permute the labels in the data while preserving the overall classification ability of the current classifier. When tested on data with a conditionally randomized labelling, the current model will achieve the same score as it does with the actual labelling, although it will misclassify different observations. This conditionally randomization test is effective when searching for models that are more adaptable to noise.

The different tests that we will contribute in this paper could be as well used in this process of building an effective induction system. However, in general our tests are not directly comparable to the conditional randomization tests of Jensen (1992) in the context of this paper. We evaluate the classifier performance on the different randomized samples, and therefore, creating data set samples that preserve such performance would only produce always p-values close to one.

The restricted randomization test that we will study in detail later, can be used for studying the importance of dependent features for the classification performance. Related to this, group variable selection is a method for finding similarities between the features (Bondell and Reich, 2008). In that approach, similar features are grouped together for decreasing the dimensionality and improving the classification accuracy. Such methods are good for clustering the features while

doing classification. However, our aim is to test whether the dependency between the features is essential in the classification and not to reduce the dimensionality and similarities, thus differing from the objective of group variable selection.

As part of the related work we should mention that there is a large amount of statistical literature about hypothesis testing (Casella and Berger, 2001). Our contribution is to use the framework of hypothesis testing for assessing the classifier performance by means of generating permutation-based p-values. How the different randomizations affect these p-values is the central question we would like to study. Also sub-sampling methods such as bootstrapping (Efron, 1979) use randomizations to study the properties of the underlying distribution, but this is not used for testing the data against some null model as we intend here.

3. Permutation Tests for Labeled Data

In this section we describe in detail two very simple permutation methods to estimate the null distribution of the error under two different null hypotheses. The questions for which the two statistical tests supply answers can be summarized as follows:

Test 1: Has the classifier found a significant class structure, that is, a real connection between the data and the class labels?

Test 2: Is the classifier exploiting a significant dependency between the features to increase the accuracy of the classification?

Note, that these tests study whether the classifier is using the described properties and not whether the plain data contain such properties. For studying the characteristics of a population represented by the data, standard statistical test could be used (Casella and Berger, 2001).

Let π be a permutation of n elements. We denote with $\pi(y)_i$ the i-th value of the vector label y induced by the permutation π. For the general case of a column vector X^j, we use $\pi(X^j)$ to represent the permutation of the vector X^j induced by π. Finally, we denote the concatenation of column vectors into a matrix by $X = [X^1, X^2, \ldots, X^m]$.

3.1 Two Simple Permutation Methods

The first permutation method is the standard permutation test used in statistics (Good, 2000). The null hypothesis assumes that the data X and the labels y are independent, that is, $p(X,y) = p(X)p(y)$. The distribution under this null hypothesis is estimated by permuting the labels in D.

Test 1 (Permute labels) *Let $D = \{(X_i, y_i)\}_{i=1}^n$ be the original data set and let π be a permutation of n elements. One randomized version D' of D is obtained by applying the permutation π on the labels, $D' = \{(X_i, \pi(y)_i)\}_{i=1}^n$. Compute the p-value as in Definition 1.*

A significant classifier for Test 1, that is, obtaining a small p-value, rejects the null hypothesis that the features and the labels are independent, meaning that there is no difference between the classes. Let us now study this by considering the following case analysis. If the original data contains a real (i.e., not a random effect) dependency between data points and labels, then: (1) a significant classifier f will use such information to achieve a good classification accuracy and this will result in a small p-value (because the randomized samples do not contain such dependency

by construction); (2) if the classifier f is not significant in the sense of Test 1 (that is, f was not able to use the existing dependency between data and labels in the original data), then the p-value would tend to be high because the error in the randomized data will be similar to the error obtained in the original data. Finally, if the original data did not contain any real dependency between data points and labels, that is, such dependency was similar to randomized data sets, then all classifiers tend to have a high p-value. However, as a nature of statistical tests, about α of the results will be incorrectly regarded as significant.

Applying randomizations on the original data is therefore a powerful way to understand how the different classifiers use the structure implicit in the data, if such structure exists. However, notice that a classifier might be using additionally some dependency structure in the data that is not checked by Test 1. Indeed, it is very often the case that the p-values obtained from Test 1 are very small on real data because a classifier is easily regarded as significant even if the class structure is weak. We will provide more evidence about this fact in the experiments.

An important point is in fact, that a good classifier can be using other types of dependency if this exists in the data, for example the dependency between the features. From this perspective, Test 1 does not generate the appropriate randomized data sets to test such hypotheses. Therefore, we propose a new test whose aim is to check for the dependency between the attributes and how classifiers use such information.

The second null hypothesis assumes that the columns in X are mutually independent inside a class, thus $p(X(c)) = p(X(c)^1) \cdots p(X(c)^m)$, where $X(c)$ represents the submatrix of X that contains all the rows having the class label $c \in \mathcal{Y}$. This can be stated also using conditional probabilities, that is, $p(X \mid y) = p(X^1 \mid y) \cdots p(X^m \mid y)$. Test 2 is inspired by the restricted randomizations from statistics (see, e.g., Good, 2000).

Test 2 (Permute data columns per class) *Let $D = \{(X_i, y_i)\}_{i=1}^n$ be the data. A randomized version D' of D is obtained by applying independent permutations to the columns of X within each class. That is:*

For each class label $c \in \mathcal{Y}$ do,

- *Let $X(c)$ be the submatrix of X in class label c, that is, $X(c) = \{X_i \mid y_i = c\}$ of size $l_c \times m$.*
- *Let $\pi_1, \ldots, \pi_m$ be m independent permutations of l_c elements.*
- *Let $X(c)'$ be a randomized version of $X(c)$ where each π_j is applied independently to the column $X(c)^j$. That is, $X(c)' = [\pi_1(X(c)^1), \ldots, \pi_m(X(c)^m)]$.*

Finally, let $X' = \{X(c)' \mid c \in Y\}$ and obtain one randomized version $D' = \{(X_i', y_i)\}_{i=1}^n$. Next, compute the p-value as in Definition 1.

Thus, a classification result can be regarded as nonsignificant with Test 2, if either the features are independent of each other inside the classes or if the classifier does not exploit the interdependency between the features. Notice that we are not testing the data but the classifier against the null hypothesis corresponding to Test 2. The classification result is significant with Test 2 only if the classifier exploits the interdependency between the features, if such interdependency exists. If the dependency is not used, there might be no reason to use a complicated classifier, as simpler and faster methods, such as Naive Bayes, could provide similar accuracy results for the same data. On

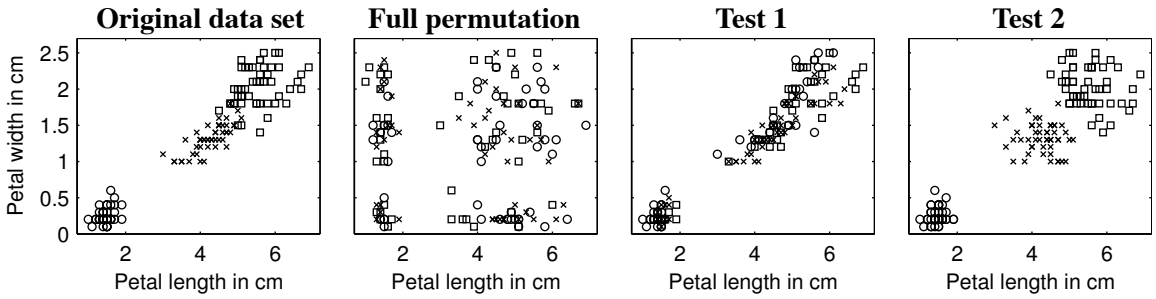


Figure 2: Scatter plots of original Iris data set and randomized versions for full permutation of the data and for Tests 1 and 2 (one sample for each test). The data points belong to three different classes denoted by different markers, and they are scattered against petal length and width in centimeters.

the other hand, this observation can lead us to find a classifier that can exploit the possibly existing dependency and thus improve the classification accuracy further, as discussed in the introduction.

There are three important properties of the permutation-based p-values and the two tests proposed here. The first one is that the number of missing values, that is, the number of entries in D that are empty because they do not have measured values, will be distributed equally across columns in the original data set D and the randomized data sets D'; this is necessary for a fair p-value computation. The second property is that the proposed permutations are always relevant regardless of the data domain, that is, values are permuted always within the same column, which does not change the domain of the randomized data sets. Finally, we have that unbalanced data sets, that is, data sets where the distribution of class labels is not uniform, remain equally unbalanced in the randomized samples.

In all, with permutation tests we obtain useful statistics about the classification result. No test is better than the other, but all provide us with information about the classifier. Each p-value is a statistic about the classifier performance; each p-value depends on the original data (whether it contains some real structure or not) and the classifier (whether it is able to use certain structure in the data or not).

In Figure 2, we give as an example one randomization for each test on the well-known Iris data set. We show here the projection of two features, before and after randomizations according to each one of the tests. For comparison, we include a test corresponding to full permutation of the data where each column is permuted separately, breaking the connection between the features and mixing the values between different classes. Note how well Test 2 has preserved the class structure compared to other tests. To provide more intuitions, in this case a very simple classifier, which predicts the class by means of one single of these two features would suffice in reaching a very good accuracy. In other words, the dependency between the two features is not significant as such, so that a more complex classifier making use of such dependency would end up having a high p-value with Test 2. We will discuss the Iris data more in the experiments.

3.2 Handling Instability of the Error

A related issue for all the above presented tests concerns the variability of the error estimate returned by a classifier. Indeed, applying the same classifier several times over the original data set D can

return different error estimates $e(f,D)$ if, for example, 10-fold cross-validation is used. So the question is, how can we ensure that the p-values given by the tests are stable to such variance?

The empirical p-value depends heavily on the correct estimation of the original classification accuracy, whereas the good estimation of the classification errors of the randomized data sets is not so important. However, exactly the same classification procedure has to be used for both the original and randomized data for the p-value to be valid. Therefore, we propose the following solution to alleviate the problem of having instable test statistic: We train the classifier on the original data r times, thus obtaining r different error estimates $E = \{e_1(f,D),\ldots,e_r(f,D)\}$ on D. Next, we obtain k randomized samples of D according to the desired null hypothesis and compute the p-value for each one of those original errors $e \in E$. We obtain therefore r different p-values by using the same k randomized data sets for each computation. We finally output the average of those r different p-values as the final empirical p-value.

Note that in total we will compute the error of the classifier $r+k$ times: r times on the original data and one time for each of the k randomized data sets. Of course, the larger the k and the larger the r, the more stable the final averaged p-value would be. A larger r decreases the variance in the final p-value due to the estimation of the classification error of the original data set whereas a larger k decreases the variance in the final p-value due to the random sampling from the null distribution. In practice, we have observed that a value of $r = 10$ and $k = 100$ produce sufficiently stable results.

This solution is closely related to calculating the statistic ρ, or calculating the test statistic U of the Wilcoxon-Mann-Whitney two-sample rank-sum test (Good, 2000). However, it is not valid to apply these approaches in our context as the r classification errors of the original data are not independent of each other. Nevertheless, the proposed solution has the same good properties as the ρ and U statistics as well as it generalizes the concept of empirical p-value to instable results.

A different solution would be to use a more accurate error estimate. For example, we could use leave-one-out cross-validation or cross-validation with 100 folds instead of 10-fold cross-validation. This will decrease the variability but increase the computation time dramatically as we need to perform the same slow classification procedure to all k randomized samples as well. However, it turns out that the stability issue is not vital for the final result; our solution produces sufficiently stable p-values in practice.

3.3 Example

We illustrate the concept of the tests by studying the small artificial example presented in the introduction in Figure 1. Consider the two data sets D_1 and D_2 given in Figure 1. The first data set D_1 was generated as follows: in the first eight rows corresponding to class +, each element is independently sampled to be x with probability 80% and o otherwise; in the last eight rows the probabilities are the other way around. Note that in the data set D_1 the features are independent given the class since, for example, knowing that $X_i^{j_1} =$ x inside class + does not increase the probability of $X_i^{j_2}$ being x. The data set D_2 was generated as follows: the first four rows contain x, the second four rows contain o, the third four rows contain x in the first four columns and o in the last four columns, and the last four rows contain o in the first four columns and x in the last four columns; finally, 10% of noise was added to the data set, that is, each x was flipped to o with probability of 10%, and vice versa.

Observe that both D_1 and D_2 have a clear separation into the two given classes, + and −. However, the structure inside the data set D_1 is much simpler than in the data set D_2. For illustration

1-Nearest Neighbor					
	Orig.	Test 1		Test 2	
Data Set	Err.	Err. (Std)	p-val.	Err. (Std)	p-val.
D_1	0.00	0.53 (0.14)	0.001	0.06 (0.06)	**0.358**
D_2	0.00	0.53 (0.14)	0.001	0.62 (0.14)	0.001

Table 1: Average error and p-value for Test 1 and Test 2 when using the 1-Nearest Neighbor classifier to data sets of Figure 1.

purposes, we analyze this with the 1-Nearest Neighbor classifier using the leave-one-out cross-validation given in Equation (1). Results for Test 1 and Test 2 are summarized in Table 1. The classification error obtained in the original data is 0.00 for both D_1 and D_2, which is expected since the data sets were generated to contain clear class structure.

First, we use the standard permutation test (i.e., permuting labels, Test 1) to understand the behavior under the null hypothesis where data points and labels are independent. We produce 1000 random permutations of the class labels for both the data sets D_1 and D_2, and perform the same leave-one-out cross-validation procedure to obtain a classification error for each randomized data set. On the randomized samples of data set D_1 we obtain an average classification error of 0.53, a standard deviation 0.14 and a minimum classification error of 0.13. For the randomized data from D_2 the corresponding values are 0.53, 0.14 and 0.19, respectively. These values result in two empirical p-values of both 0.001 on both the data sets D_1 and D_2. Thus, we can say that the classifiers are significant under the null hypothesis that data and labels are independent. That is, the connection between the data and the class labels is real in both data sets and the 1-Nearest Neighbor classifier is able to find that connection in both data sets, resulting into a good classification accuracy.

However, it is easy to argue that the results of Test 1 do not provide much information about the classifier performance. Actually the main problem of Test 1 is that p-values tend to be always very low as the null hypothesis is typically easy to reject. To get more information of the properties of the classifiers, we study next the performance of the classifiers by taking into account the inner structure of data sets D_1 and D_2 by applying Test 2. Again, we produce 1000 random samples of the data sets D_1 and D_2 by permuting each column separately inside each class. The same leave-one-out cross-validation procedure is performed for the randomized samples, obtaining for the data set D_1 the average classification error of 0.06, standard deviation of 0.06 and a minimum value of 0.00. For the data set D_2 the corresponding values are 0.62, 0.14 and 0.19, respectively. Therefore, under Test 2 the empirical p-values are 0.358 for the data set D_1 and 0.001 for the data set D_2.

We can say that, for Test 2, the 1-Nearest Neighbor classifier is significant for data set D_2 but not for data set D_1. Indeed, the data set D_1 was generated so that the features are independent inside the classes, and hence, the good classification accuracy of the algorithm on D_1 is simply due to different value distributions across the classes. Note, however, that none of the features in the data set D_1 is sufficient alone to correctly classify all the samples due to the noise in the data set. Thus using a combination of multiple features for classification is necessary for obtaining a good accuracy, even though the features are independent of each other. For data set D_2 we have that the dependency between the columns inside the classes is essential for the good classification result, and in this case, the 1-Nearest Neighbor classifier has been able to exploit that information.

4. Analysis

In this section we analyze the properties of the tests and demonstrate the behavior of the different p-values on simulated data. First, we state the relationships between the different sets of permutations.

4.1 Connection between Test 1 and Test 2

Remember that the random samples from Test 1 are obtained by permuting the class labels and the samples from Test 2 by permuting the features inside each class. To establish a connection between these randomizations, we study the randomization where each data column is permuted separately, regardless of the class label. This corresponds to the full permutation presented in Figure 2 in Section 3.1 for Iris data set. It breaks the connection between the features, and furthermore, between the data and the class labels. The following result states the relationship between Test 1, Test 2 and the full permutation method.

Proposition 2 *Let $\Pi_l(D)$, $\Pi_c(D)$, $\Pi_{cc}(D)$ be the sets of all possible randomized data sets obtained from D via permuting labels (Test 1), permuting data columns (full permutation), or permuting data columns inside class (Test 2), respectively. The following holds,*

(1) $\Pi_l(D) \subset \Pi_c(D)$

(2) $\Pi_{cc}(D) \subset \Pi_c(D)$

(3) $\Pi_l(D) \neq \Pi_{cc}(D)$

Note that $\Pi_l(D)$, $\Pi_c(D)$ and $\Pi_{cc}(D)$ refer to sets of data matrices. Therefore, we have that permuting the data columns is the randomization method producing the most diverse samples, while permuting labels (Test 1) and permuting data within class (Test 2) produce different randomized samples.

Actually, the relationship stated by Proposition 2 implies the following property: the p-value obtained by permuting the data columns is typically smaller than both the p-values obtained from Test 1 and Test 2. The reason is that all the randomized data sets obtained by Test 1 and Test 2 can also be obtained by permuting data columns and the additional randomized data sets obtained by permuting the columns are, in general, even more random. Theoretically, permuting the data columns is a combination of Test 1 and Test 2, and thus, it is not a useful test. In practice, we have observed that the p-value returned by permuting the data columns is very close to the p-value of Test 1, which tends to be much smaller than the p-value of Test 2.

Considering Proposition 2, it makes only sense to restrict the randomization to classes by using Test 2, whenever Test 1 has produced a small p-value. That is, it is only reasonable to study whether the classifier uses feature dependency in separating the classes if it has found a real class structure.

4.2 Behavior of the Tests

To understand better the behavior of the tests, we study generated data where correlation is used as the dependency between the features. Consider the following simulated data, inspired by the data used by Golland et al. (2005): 100 data points are generated from two-dimensional normal distribution with mean vector (1,0), unit variances and covariance $\rho \in [-1,1]$. Another 100 data points are generated from similar normal distribution with mean $(-1,0)$, unit variances and same

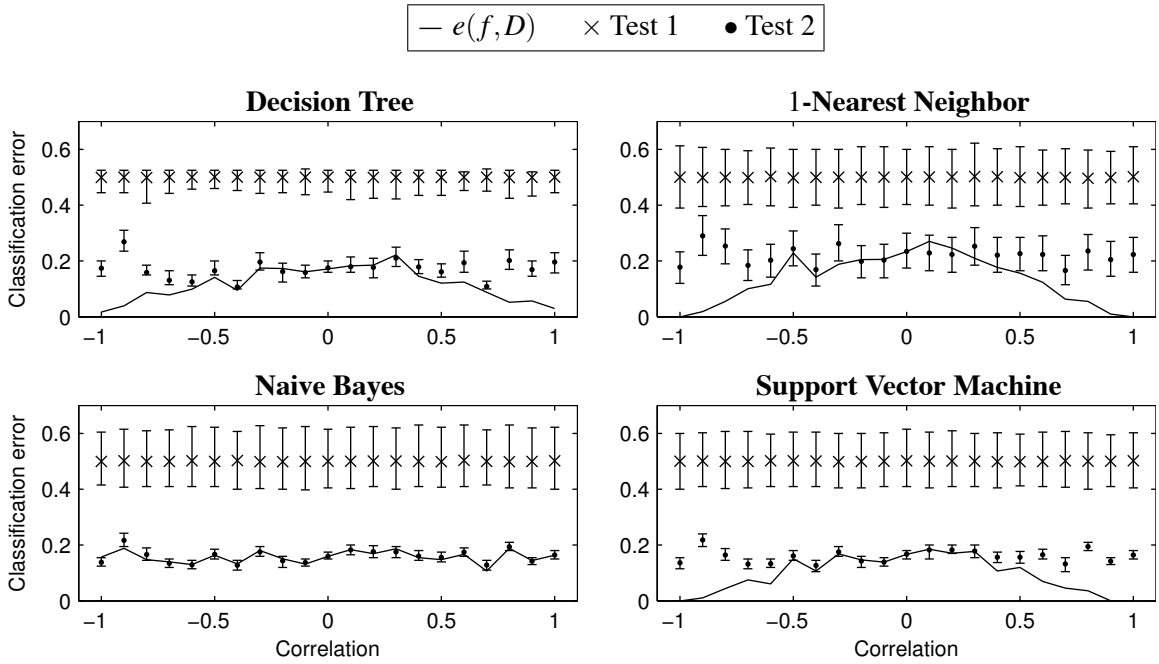


Figure 3: Average values of stratified 10-fold cross-validation error (y-axis) for varying values of correlation between the features per class (x-axis). The solid line shows the error on the original data, and symbols $\times$ and $\bullet$ represent the average of the error on 1000 randomized samples obtained from Test 1 and from Test 2, respectively. Each average of the error on the randomized samples $\times$ and $\bullet$ is depicted together with the $[1\%, 99\%]$-deviation bar. If the solid line falls below the bars the null hypothesis associated to the test is rejected; if the solid line crosses inside or above the bars the null hypothesis cannot be rejected with significance level $\alpha = 0.01$.

covariance ρ. The first 100 samples are assigned with class label $y = +1$ with probability $1-t$ and $y = -1$ with probability t. For the other 100 samples the probabilities are the opposite. The probability $t \in [0, 0.5]$ represents the noise level. When $t = 0.5$, there is no class structure at all. Note that the correlation between the features improves the class separation: if the correlation $\rho = 1$ and the noise $t = 0$, we have that the class $y = x_1 - x_2$ where x_1, x_2 are the values of the first and second features, respectively.

For these data sets (with varying parameters of noise and correlation) we use as an error estimate the stratified 10-fold cross-validation error. We study the behavior of four classifiers: 1-Nearest Neighbor, Decision Tree, Naive Bayes and Support Vector Machine. We use Weka 3.6 data mining software (Witten and Frank, 2005) with the default parameters of the implementations of those classification algorithms. The Decision Tree classifier is similar to C4.5 algorithm, and the default kernel used with Support Vector Machine is linear. Tuning the parameters of these algorithms is not in the scope of this paper; our objective is to show the behavior of the discussed p-values for some selected classifiers.

Figure 3 shows the behavior of the classifiers on data sets without class noise, $t = 0$, and with the correlation ρ between features inside classes varying from -1 (negative correlation) to 1 (positive correlation). The solid line corresponds to $e(f,D)$, that is, the error of the classifier in the original

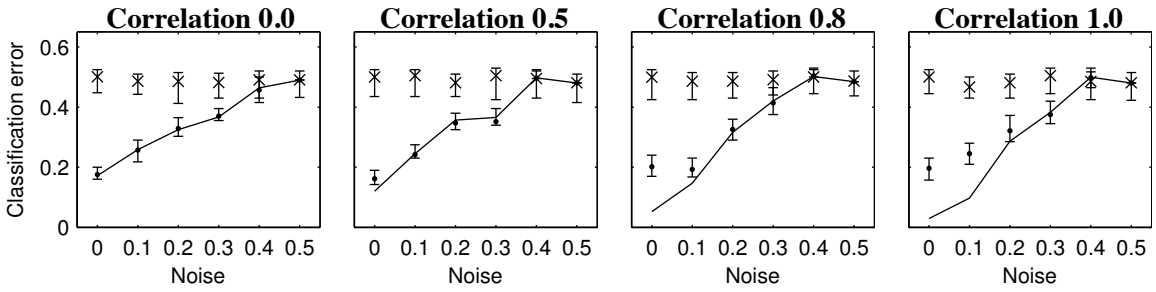


Figure 4: Average values of stratified 10-fold cross-validation error (y-axis) for the Decision Tree classifier when noise varies on the original data set (x-axis) with four fixed correlation values between the features inside the classes. The solid line shows the error on the original data, and symbols $\times$ and $\bullet$ show the average error on 1000 randomized samples from Test 1 and Test 2, respectively. Each average of the error on the randomized samples $\times$ and $\bullet$ is depicted together with the $[1\%, 99\%]$-deviation bar below which the associated null hypothesis is rejected with significance level $\alpha = 0.01$.

data. The symbols "$\times$" and "$\bullet$" represent the average error obtained by the classifier on 1000 randomized samples from Test 1 and Test 2, respectively. When the solid line of $e(f,D)$ falls below the $[1\%, 99\%]$-deviation bars, the corresponding associated null hypothesis is rejected with significance level $\alpha = 0.01$. Actually, the correspondence between the confidence intervals and hypotheses testing is only approximately true since the definition of empirical p-value contains the addition of 1 in both the numerator and denominator. However, the practical difference is negligible.

First, note that the Decision Tree, 1-Nearest Neighbor and Support Vector Machine classifiers have been able to exploit the dependency between the features, that is, the classification error goes to zero when there is either a high positive or negative correlation between the features. However, with Naive Bayes classifier the classification error seems to be independent of the correlation between the features.

For all classifiers we observe that the null hypothesis associated to Test 1 (i.e., labels and data are independent) is always rejected. Thus the data contains a clear class structure as expected since there exists no class noise in the data. All classifiers are therefore significant under Test 1.

Another expected observation is that the null hypothesis for Test 2 (i.e., features are independent within class) tends to be rejected as the magnitude of the correlation between features increases. That is, the correlation is useful in classifying the data. When the magnitude of the correlation is larger than approximately 0.4, the Decision Tree, Nearest Neighbor and Support Vector Machine classifiers reject the null hypothesis. Thus these classifiers produce significant results under Test 2 when the features are highly correlated.

Finally, observe the behavior of Naive Bayes classifier for Test 2: the null hypothesis can never be rejected. This is because Naive Bayes classifier explicitly assumes by default that the features are independent, thus it always performs similarly on the original data and the randomized data sets, which results in a very high p-value. Naive Bayes classifier is an example of such classifiers which are not able to use the dependency between the features at all. Thus applying Test 2 for Naive Bayes classifier will practically always produce a high p-value irrespective of the data.

Finally, Figure 4 shows the behavior of the Decision Tree classifier when the noise $t \in [0, 0.5]$ is increased on the x-axis. We also vary the correlation ρ between the features per class and show the results on four cases: zero correlation, 0.5, 0.8 and total correlation. We observe that as the noise increases the p-values tend to be larger. Therefore, it is more difficult to reject the null hypothesis on very noisy data sets, that is, when the class structure is weak. This is true for both Test 1 and Test 2. However, Test 1 rejects the null hypothesis even if there is 30% of noise. This supports the fact already observed in related literature (Golland et al., 2005), that even a weak class structure is easily regarded as significant with Test 1. Compared to this, Test 2 gives more conservative results.

4.3 Power Analysis of Test 2

The *power* of a statistical test is the probability that the test will reject the null hypothesis when the alternative hypothesis is true. The power of the test depends on how much or how clearly the null hypothesis is false. For example, in our case with Test 2, a classifier may rely solely on a strong dependency structure between some specific features in the classification, or it may use a weak feature dependency to slightly improve the classification accuracy. Rejecting the null hypothesis of Test 2 is much easier in the former than in the latter case. Note, however, that a strong dependency between the features is not always useful in separating the classes, as seen in Figure 2 with Iris data set. So, the question with Test 2 is whether the classifier is exploiting some of the dependency structure between the features in the data and how important such feature dependency is for the classification of the data.

In general, the power of the test can only be analyzed in special cases. Nevertheless, such analysis can give some general idea of the power the test. Next, we present a formal power analysis in the particular case where we vary the correlation between the features that is useful in separating the classes from each other. Note, however, that there exist also other types of dependency than correlation. The amount of correlation is just easy to measure, thus being suitable for formal power analysis.

We present the power analysis on similar data as studied in Section 4.2. The results in the previous subsection can be seen as informal power analysis. In summary, we observed that when the magnitude of the correlation in the data studied in Section 4.2 was larger than about 0.5 and the classifier was exploiting the feature dependency, that is, a classifier different from Naive Bayes, Test 2 was able to reject the null hypothesis. However, based on the data it is clear that even smaller correlations increased the class separation and were helpful in classifying the data but Test 2 could not regard such improvement as significant. The following analysis supports these observations.

Let the data set X consist of n points with two features belonging to two classes, $+1$ and -1. Let a point $x \in X$ be in class $y = +1$ with probability 0.5 and in class $y = -1$ with probability 0.5. Let the point $x \in X$ be sampled from two-dimensional normal distribution with mean $(0,0)$, unit variances and covariance $y\rho$ where $\rho \in [0,1]$ is a given parameter. Thus, in the first class, $y = +1$, the correlation between the two features is positive and in the second class, $y = -1$, it is negative. Compared to the data sets in Section 4.2, now the covariance changes between the classes, not the mean vector. An optimal classifier assigns a point $x \in X$ to class $y = +1$ if $x_1 x_2 > 0$ and to class $y = -1$ if $x_1 x_2 < 0$, where x_i is the i-th feature of the vector x.

The null hypothesis of Test 2 is that the classifier is not exploiting the dependency between the features in classification. To alleviate the power analysis, we assume that the classifier is able to find the optimal classification, that is, it assigns the point x to class $\text{sgn}(x_1 x_2)$ where $\text{sgn}(\cdot)$ is

the signum function. If the classifier is not optimal, it will just decrease the power of the test. The nonoptimality of the classifier could be taken into account by introducing a probability t for reporting a nonoptimal class label; this approach is used in the next subsection for power analysis of Test 1 but is left out here for simplicity in the analysis. Under this optimality scenario, the probability of correctly classifying a sample is

$$\begin{aligned}\Pr(\mathrm{sgn}(x_1x_2)=y) &= \frac{1}{2}\Pr(x_1x_2>0 \mid y=+1)+\frac{1}{2}\Pr(x_1x_2<0 \mid y=-1)\\ &= \Pr(x_1x_2>0 \mid y=+1) = 2\int_0^\infty\int_0^\infty \Pr(x_1,x_2)\,\mathrm{d}x_1\mathrm{d}x_2\\ &= 2\int_0^\infty\int_0^\infty \frac{1}{2\pi\sqrt{1-\rho^2}}\exp\left[-\frac{x_1^2-2\rho x_1x_2+x_2^2}{2(1-\rho^2)}\right]\mathrm{d}x_1\mathrm{d}x_2\\ &= \frac{1}{2}+\frac{1}{\pi}\arcsin\rho, \end{aligned} \tag{2}$$

where $\Pr(x_1,x_2)$ is just the standardized bivariate normal distribution. The null hypothesis corresponds to the case where the correlation parameter is zero, $\rho=0$, that is, no feature dependency exists. In that case, the probability of correctly classifying a sample is $1/2$.

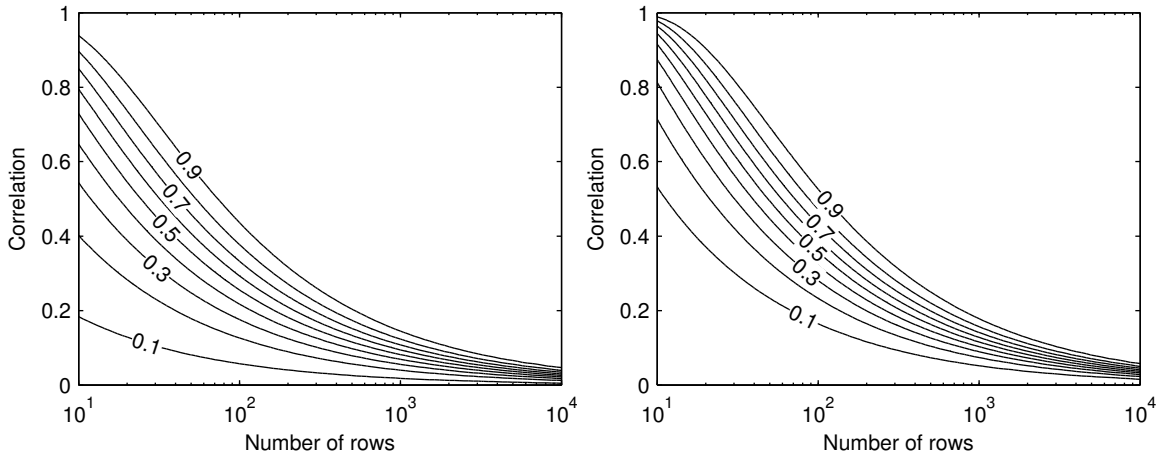
In our randomization approach, we are using classification error as the test statistic. Since we assume that the optimal classifier is given, we use all the n points of the data set X for testing the classifier and calculating the classification error. Under the null hypothesis H_0 and under the alternative hypothesis H_1 of Test 2, the classification errors $e(f \mid H_0)$ and $e(f \mid H_1)$ are distributed as follows:

$$\begin{aligned} n\cdot e(f \mid H_0) &\sim \mathrm{Bin}\left(n,\frac{1}{2}\right) \approx \mathcal{N}\left(\frac{n}{2},\frac{n}{4}\right),\\ n\cdot e(f \mid H_1) &\sim \mathrm{Bin}\left(n,\frac{1}{2}-\frac{1}{\pi}\arcsin\rho\right) \approx \mathcal{N}\left(\frac{n}{2}-\frac{n}{\pi}\arcsin\rho,\frac{n}{4}-\frac{n}{\pi^2}\arcsin^2\rho\right), \end{aligned}$$

where $\frac{1}{2}-\frac{1}{\pi}\arcsin\rho$ is the probability of incorrectly classifying a sample by Equation (2). The normal approximation $\mathcal{N}(np,np(1-p))$ of a binomial distribution $\mathrm{Bin}(n,p)$ holds with good accuracy when $np>5$ and $n(1-p)>5$. In our case, the approximation is valid if $n(\frac{1}{2}-\frac{1}{\pi}\arcsin\rho)>5$. This holds, for example, if $n\geq 20$ and $\rho\leq 0.7$.

Now the power of Test 2 for this generated data is the probability of rejecting the null hypothesis H_0 of $\rho=0$ with significance level α when the alternative hypothesis H_1 is that the correlation $\rho>0$. Note that we are implicitly assuming that the classifier is optimal, that is, we are excluding the classifier quality from the power analysis. Thus, the power is the probability that $e(f \mid H_1)$ is smaller than $1-\alpha$ of the errors $e(f \mid H_0)$ under the alternative hypothesis H_1:

$$\begin{aligned}\mathrm{Power} &= \Pr\left(e(f \mid H_1) < F^{-1}_{e(f|H_0)}(\alpha)\right)\\ &\approx \Pr\left(\frac{1}{2}-\frac{1}{\pi}\arcsin\rho+\sqrt{\frac{1}{4n}-\frac{1}{n\pi^2}\arcsin^2\rho}\cdot Z<\frac{1}{2}+\frac{1}{2\sqrt{n}}\Phi^{-1}(\alpha)\right)\\ &= \Phi\left(\frac{2\sqrt{n}\arcsin\rho+\pi\Phi^{-1}(\alpha)}{\sqrt{\pi^2-4\arcsin^2\rho}}\right), \end{aligned} \tag{3}$$



(a) $\alpha = 0.05$ (b) $\alpha = 0.01$

Figure 5: Contour plots of the statistical power of Test 2 as a function of the number of rows n in the generated data set and the correlation parameter ρ. Each solid line corresponds to a constant value of the power that is given on top of the contour. The power values are calculated by Equation (3) for two different values of significance level α.

where $F_{e(f|H_0)}$ is the cumulative distribution function of $e(f \mid H_0)$, Z is a random variable following standard normal distribution and Φ is the cumulative distribution function of the standard normal distribution. Note that we are using exact p-value instead of empirical p-value, effectively leaving out the influence of variance by using k randomized samples; see Fay et al. (2007) for analysis of resampling risk of using k samples. However, this has little effect to the power of the test. When the correlation $\rho = 0$, the power is α, that is, when the null hypothesis is true, it is rejected incorrectly about α of the times. Therefore, α is really the significance level of the tests.

In Figure 5 we present contour plots of the statistical power in Equation (3) for different values of the two varying parameters. As expected, the higher the correlation ρ and the number of rows n are, the higher the statistical power of Test 2 is. For example, if the data set contains about 1000 rows, we can infer with 90% probability that the classifier is exploiting the feature dependency of approximately a correlation of 0.2 in the data. The results are also in line with the results from Section 4.2 although the studied data sets are slightly different. When the significance level used is $\alpha = 0.01$ we can infer that the classifier is exploiting the feature dependency of correlation larger than 0.4 approximately 90% of the times when the data set has 200 rows.

Notice that if we had not considered an "optimal" classifier, that is, if we had introduced a probability t of assigning each observation to the incorrect label, then Equation (3) would depend on three parameters. In that case, the higher t, the smaller is the power of the test; however, for a fixed t we still would observe the same behaviour as in the contourplots above: the higher the correlation ρ and the larger the n, the higher is the statistical power of Test 2. The error parameter t is taken into account in the next section, where the power analysis of Test 1 does not depend on ρ between the features.

4.4 Power Analysis of Test 1

Let the data set X consist of n observations belonging to two different classes with equal probability. We assume that we have a classifier f whose error rate is $t \in [0,1]$, that is, the classifier assigns each observation to the correct class with probability $1-t$. Another way to see this is that the classifier f is optimal but the original class label of each point is erroneous with probability t. We perform power analysis of Test 1 for this general form of data.

Note that the results in Section 4.2 can be seen as informal power analysis of Test 1 on similar setting as studied here. The results in Figure 4 can be summarized as follows. When the error rate was smaller than $t < 0.4$, Test 1 was able to reject the null hypotheses. Note, however, that the error rate t used in this section is not directly comparable to the error rate used in Section 4.2.

The power analysis of Test 1 proceeds similarly as in the previous subsection for Test 2. Under the null hypothesis H_0 and under the alternative hypothesis H_1 of Test 1, the classification errors $e(f \mid H_0)$ and $e(f \mid H_1)$ are distributed as follows:

$$
\begin{aligned}
n \cdot e(f \mid H_0) &\sim \text{Bin}\left(n, \frac{1}{2}\right) \approx \mathcal{N}\left(\frac{n}{2}, \frac{n}{4}\right), \\
n \cdot e(f \mid H_1) &\sim \text{Bin}\,(n,t) \approx \mathcal{N}(nt, nt(1-t)).
\end{aligned}
$$

The null hypothesis H_0 assumes that there is no connection between the data and the class labels thus the probability of incorrect classification is $1/2$ as the classes are equally probable. Note that the null hypothesis corresponds to the case where the error rate of the classifier f is $t = 1/2$.

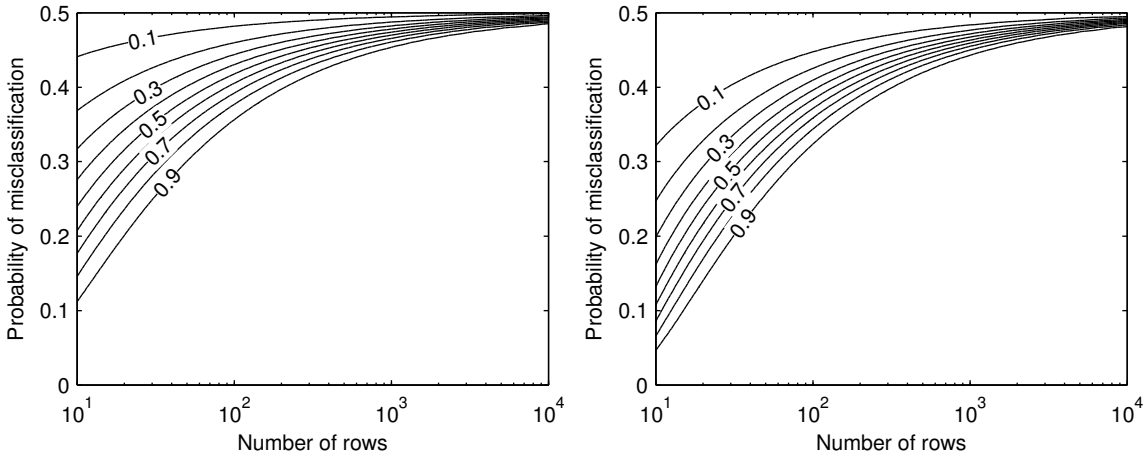
Now the power of Test 1 is the probability of rejecting the null hypothesis H_0 with significance level α when the alternative hypothesis H_1 is true, that is,

$$
\begin{aligned}
\text{Power} &= \Pr\left(e(f \mid H_1) < F^{-1}_{e(f|H_0)}(\alpha)\right) \\
&\approx \Pr\left(t + \sqrt{\frac{t(1-t)}{n}}\, Z < \frac{1}{2} + \frac{1}{2\sqrt{n}}\, \Phi^{-1}(\alpha)\right) \\
&= \Phi\left(\frac{(1-2t)\sqrt{n} + \Phi^{-1}(\alpha)}{2\sqrt{t(1-t)}}\right), \qquad (4)
\end{aligned}
$$

where the same notation as in the previous subsection is used. First, note that when the null hypothesis is true, that is, $t = 1/2$, the power of Test 1 calculated by Equation (4) equals the significance level α as it should.

In Figure 6 we present contour plots of the statistical power of Test 1 calculated by Equation (4) for different values of parameters. As expected, when the number of observations n increases or the error rate t decreases, the power increases. Furthermore, the larger the significance level α is, the larger the power of Test 1 is. When the parameter values are $\alpha = 0.01$, $n = 200$ and $t = 0.4$, the power of Test 1 is about 0.7 that is comparable to the results in Section 4.2.

In this section, we analyzed the behaviour and the power of the tests. Note that although we used correlation as the only type of dependency between features in this section, there exist also other forms of dependency that the classifier can exploit. As conclusions from the power analysis, the more rows the data set has, the easier we can infer that the classifier is using the feature dependency or some other properties in the data.



(a) $\alpha = 0.05$ (b) $\alpha = 0.01$

Figure 6: Contour plots of the statistical power of Test 1 as a function of the number of rows n in the generated data set and the probability of misclassification t. Each solid line corresponds to a constant value of the power that is given on top of the contour. The power values are calculated by Equation (4) for two different values of significance level α.

5. Empirical Results

In this section, we give extensive empirical results on 33 various real data sets from UCI machine learning repository (Asuncion and Newman, 2007). Basic characteristics of the data sets are described in Table 2. The data sets are divided into three categories based on their size: small, medium and large. Some data sets contain only nominal or numeric features whereas some data sets contain both kind of features (mixed). About one-third of the data sets contain also missing values. Notice that in most data sets the features are measured in different scales, thus it is not sensible to swap the values between different features. This justifies why it is only reasonable to consider column-wise permutations, and why some recent data mining randomization methods (Gionis et al., 2007; Ojala et al., 2009; Chen et al., 2005) are not generally applicable in assessing classification results.

In the experiments we use Weka 3.6 data mining software (Witten and Frank, 2005) that contains open source Java implementations of many classification algorithms. We use four different types of classification algorithms with the default parameters: Decision Tree, Naive Bayes, 1-Nearest Neighbor and Support Vector Machine classifier. The Decision Tree classifier is similar to C4.5 algorithm. The default kernel used with Support Vector Machine is linear. Missing values and the combination of nominal and numerical values are given as such as the input for the classifiers; the default approaches in Weka of the classifiers are used to handle these cases. Notice that tuning the parameters of these algorithms is not in the scope of this paper; our objective is to show the behavior of the discussed p-values for some selected classifiers on various data sets.

We use different classification procedures and the number of randomized data sets for each of the different size categories of the data sets (small, medium and large). For small data sets, we use stratified 10-fold cross-validation error as the statistic and 1000 randomized data sets for calculating the empirical p-values. For medium-sized data sets, we use the same stratified 10-fold cross-validation error and 100 randomized data sets. Finally, for large data sets, we divide the data

	Data Set	Rows	Features	Classes	Missing	Domain
Small	Audiology	226	70	24	2.0%	nominal
	Autos	205	25	6	1.2%	mixed
	Breast	286	9	2	0.3%	nominal
	Glass	214	9	6	No	numeric
	Hepatitis	155	19	2	5.7%	mixed
	Ionosphere	351	34	2	No	numeric
	Iris	150	4	3	No	numeric
	Lymph	148	18	4	No	mixed
	Promoters	106	57	2	No	nominal
	Segment	210	19	7	No	numeric
	Sonar	208	60	2	No	numeric
	Spect	267	22	2	No	nominal
	Tumor	339	17	21	3.9%	nominal
	Votes	435	16	2	5.6%	nominal
	Zoo	101	17	7	No	mixed
Medium	Abalone	4177	8	28	No	mixed
	Anneal	898	38	5	65.0%	mixed
	Balance	625	4	3	No	numeric
	Car	1728	6	4	No	nominal
	German	1000	20	2	No	mixed
	Mushroom	8124	22	2	1.4%	nominal
	Musk	6598	166	2	No	numeric
	Pima	768	8	2	No	numeric
	Satellite	6435	36	6	No	numeric
	Spam	4601	57	2	No	numeric
	Splice	3190	60	3	No	nominal
	Tic-tac-toe	958	9	2	No	nominal
	Yeast	1484	8	10	No	numeric
Large	Adult	48842	15	2	0.9%	mixed
	Chess	28056	6	18	No	mixed
	Connect-4	67557	42	3	No	nominal
	Letter	20000	16	26	No	numeric
	Shuttle	58000	9	7	No	numeric

Table 2: Summary of 33 selected data sets from UCI machine learning repository (Asuncion and Newman, 2007). The data sets are divided into three categories based on their size: small, medium and large.

set into training set with 10 000 random rows and to test set with the rest of the rows. We use 100 randomized data sets for calculating the empirical p-values with large data sets. The reason for the smaller number of randomized samples for medium and large data sets is mainly computation time. However, 100 samples is usually enough for statistical inference. Furthermore, as seen in Section 4 the power of the tests is greater when the data sets have more rows, that is, with large data sets it is easier to reject the null-hypotheses, supporting the need of fewer randomized samples in hypothesis testing.

Since the original classification error is not a stable result due to the randomness in training the classifier and dividing the data set into test and train data, we perform the same classification procedure ten times for the original data sets and calculate an empirical p-value for each of the ten results. This was described in Section 3.2. We give the average value of these empirical p-values as well as the average value and the standard deviation of the original classification errors.

As we are testing multiple hypotheses simultaneously, we need to correct for multiple comparisons. We apply the approach by Benjamini and Hochberg (1995) to control the false discovery rate (FDR), that is, the expected proportion of results incorrectly regarded as significant. In the experiments, we restrict the false discovery rate below $\alpha = 0.05$ separately for Test 1 and Test 2. In the Benjamini-Hochberg approach, if $p_1, \ldots, p_m$ are the original empirical p-values in increasing order, the results $p_1, \ldots, p_l$ are regarded as significant where l is the largest value such that $p_l \leq \frac{l}{m}\alpha$.

The significance testing results for the Decision Tree classifier are given in Table 3, for Naive Bayes in Table 4, for 1-Nearest Neighbor classifier in Table 5 and finally for Support Vector Machine classifier in Table 6. The mean and the standard deviation of the 10 original classification errors are given as well as the mean and standard deviation of the errors on the 1000 or 100 randomized samples with Test 1 and Test 2. The empirical p-values corresponding to nonsignificant results, when the false discovery rate is restricted below 0.05, are in boldface in the tables. With all classifiers, the largest significant empirical p-value was 0.01. The smallest non-significant p-values were 0.03 with Decision Tree and 1-Nearest Neighbor classifiers, 0.08 with Naive Bayes classifier and 0.19 with Support Vector Machine classifier.

The results for the traditional permutation method Test 1 show that the classification errors with most data sets are regarded as significant. These results show that the data sets contain clear class structure. However, they do not give any additional insight for understanding the class structure in the data sets.

There are two reasons why the simple permutation test, Test 1, regards the class structure of the data sets as significant. Firstly, most of the data sets that are publicly available, as all the data sets used in this paper, have already passed some quality checks, that is, someone has already found some interesting structure in them. Secondly, and as a more important reason, the traditional permutation tests easily regard the results as significant even if there is only a slight class structure present because in the corresponding permuted data sets there is no class structure, especially if the original data set is large.

Furthermore, the few results which were regarded as nonsignificant with Test 1 are with such classifiers that have not performed well on the data. That is, the other classifiers have produced smaller classification errors on the same data sets, and, in contrast, these results are regarded as significant.

Next, we consider the results for permuting the features inside each class, that is, Test 2. The results show that there are actually now almost equal amount of nonsignificant and significant results with respect to Test 2. This means that in many data sets the original structure inside the classes is

Decision Tree

		Original	Test 1		Test 2	
	Data Set	Err. (Std)	Err. (Std)	p-val.	Err. (Std)	p-val.
Small	Audiology	0.22 (0.01)	0.82 (0.03)	0.001	0.23 (0.02)	**0.482**
	Autos	0.19 (0.01)	0.76 (0.04)	0.001	0.38 (0.04)	0.001
	Breast	0.26 (0.01)	0.30 (0.00)	0.001	0.29 (0.02)	**0.116**
	Glass	0.33 (0.02)	0.72 (0.03)	0.001	0.34 (0.03)	**0.457**
	Hepatitis	0.22 (0.02)	0.23 (0.02)	**0.319**	0.15 (0.03)	**0.955**
	Ionosphere	0.10 (0.01)	0.38 (0.02)	0.001	0.07 (0.01)	**0.964**
	Iris	0.05 (0.01)	0.67 (0.03)	0.001	0.05 (0.01)	**0.765**
	Lymph	0.22 (0.02)	0.51 (0.05)	0.001	0.23 (0.04)	**0.437**
	Promoters	0.21 (0.04)	0.50 (0.06)	0.002	0.22 (0.05)	**0.377**
	Segment	0.13 (0.02)	0.86 (0.03)	0.001	0.17 (0.02)	**0.132**
	Sonar	0.27 (0.02)	0.49 (0.03)	0.001	0.27 (0.03)	**0.507**
	Spect	0.19 (0.01)	0.22 (0.01)	0.004	0.15 (0.02)	**0.966**
	Tumor	0.58 (0.01)	0.82 (0.02)	0.001	0.60 (0.02)	**0.138**
	Votes	0.03 (0.00)	0.42 (0.02)	0.001	0.03 (0.01)	**0.791**
	Zoo	0.07 (0.01)	0.64 (0.03)	0.001	0.07 (0.01)	**0.593**
Medium	Abalone	0.79 (0.01)	0.89 (0.00)	0.01	0.67 (0.01)	**1.00**
	Anneal	0.07 (0.01)	0.24 (0.00)	0.01	0.13 (0.01)	0.01
	Balance	0.22 (0.01)	0.55 (0.02)	0.01	0.29 (0.02)	0.01
	Car	0.08 (0.00)	0.30 (0.00)	0.01	0.26 (0.01)	0.01
	German	0.29 (0.01)	0.32 (0.01)	0.01	0.28 (0.01)	**0.66**
	Mushroom	0.00 (0.00)	0.50 (0.01)	0.01	0.01 (0.00)	0.01
	Musk	0.03 (0.00)	0.16 (0.00)	0.01	0.09 (0.00)	0.01
	Pima	0.25 (0.01)	0.35 (0.01)	0.01	0.24 (0.01)	**0.67**
	Satellite	0.14 (0.00)	0.81 (0.00)	0.01	0.07 (0.00)	**1.00**
	Spam	0.07 (0.00)	0.40 (0.00)	0.01	0.06 (0.00)	**1.00**
	Splice	0.06 (0.00)	0.60 (0.01)	0.01	0.07 (0.01)	0.01
	Tic-tac-toe	0.15 (0.01)	0.36 (0.01)	0.01	0.30 (0.01)	0.01
	Yeast	0.44 (0.01)	0.76 (0.01)	0.01	0.47 (0.01)	**0.03**
Large	Adult	0.00 (0.00)	0.24 (0.00)	0.01	0.00 (0.00)	**1.00**
	Chess	0.46 (0.00)	0.89 (0.00)	0.01	0.77 (0.00)	0.01
	Connect-4	0.25 (0.00)	0.34 (0.00)	0.01	0.33 (0.00)	0.01
	Letter	0.16 (0.01)	0.96 (0.00)	0.01	0.38 (0.01)	0.01
	Shuttle	0.00 (0.00)	0.21 (0.00)	0.01	0.01 (0.00)	0.01

Table 3: Classification errors and empirical p-values obtained with Decision Tree classifier for Test 1 and Test 2. The empirical p-values are calculated over 1000 randomized samples for small data sets and over 100 randomized samples for medium and large data sets. Classification on the original data is repeated ten times. In the table, the average values and standard deviations of the classification errors are given. Bold p-values correspond to nonsignificant results when the false discovery rate is restricted below 0.05 with Benjamini and Hochberg (1995) approach.

Naive Bayes

		Original	Test 1		Test 2	
	Data Set	Err. (Std)	Err. (Std)	p-val.	Err. (Std)	p-val.
Small	Audiology	0.27 (0.00)	0.79 (0.03)	0.001	0.26 (0.01)	**0.869**
	Autos	0.43 (0.01)	0.79 (0.04)	0.001	0.22 (0.02)	**1.000**
	Breast	0.27 (0.01)	0.33 (0.02)	0.001	0.24 (0.02)	**0.959**
	Glass	0.52 (0.02)	0.81 (0.05)	0.001	0.45 (0.02)	**0.994**
	Hepatitis	0.16 (0.01)	0.30 (0.05)	0.001	0.09 (0.02)	**1.000**
	Ionosphere	0.17 (0.00)	0.46 (0.03)	0.001	0.01 (0.01)	**1.000**
	Iris	0.05 (0.01)	0.67 (0.05)	0.001	0.01 (0.01)	**0.999**
	Lymph	0.16 (0.01)	0.53 (0.05)	0.001	0.11 (0.02)	**0.995**
	Promoters	0.08 (0.01)	0.50 (0.06)	0.001	0.07 (0.02)	**0.746**
	Segment	0.21 (0.01)	0.86 (0.03)	0.001	0.13 (0.01)	**1.000**
	Sonar	0.32 (0.01)	0.50 (0.04)	0.001	0.13 (0.02)	**1.000**
	Spect	0.21 (0.01)	0.25 (0.03)	**0.077**	0.07 (0.01)	**1.000**
	Tumor	0.50 (0.01)	0.81 (0.02)	0.001	0.49 (0.02)	**0.751**
	Votes	0.10 (0.00)	0.44 (0.02)	0.001	0.00 (0.00)	**1.000**
	Zoo	0.03 (0.00)	0.81 (0.05)	0.001	0.03 (0.01)	**0.541**
Medium	Abalone	0.76 (0.00)	0.88 (0.01)	0.01	0.56 (0.01)	**1.00**
	Anneal	0.35 (0.01)	0.36 (0.04)	**0.65**	0.31 (0.01)	**1.00**
	Balance	0.09 (0.00)	0.54 (0.02)	0.01	0.24 (0.01)	0.01
	Car	0.14 (0.00)	0.30 (0.00)	0.01	0.24 (0.01)	0.01
	German	0.25 (0.00)	0.33 (0.01)	0.01	0.23 (0.01)	**1.00**
	Mushroom	0.04 (0.00)	0.50 (0.01)	0.01	0.00 (0.00)	**1.00**
	Musk	0.16 (0.00)	0.34 (0.06)	0.01	0.02 (0.00)	**1.00**
	Pima	0.24 (0.00)	0.37 (0.01)	0.01	0.22 (0.01)	**0.99**
	Satellite	0.20 (0.00)	0.80 (0.02)	0.01	0.00 (0.00)	**1.00**
	Spam	0.20 (0.00)	0.49 (0.05)	0.01	0.10 (0.00)	**1.00**
	Splice	0.05 (0.00)	0.53 (0.01)	0.01	0.03 (0.00)	**1.00**
	Tic-tac-toe	0.30 (0.00)	0.35 (0.01)	0.01	0.28 (0.01)	**1.00**
	Yeast	0.42 (0.00)	0.71 (0.01)	0.01	0.42 (0.01)	**0.36**
Large	Adult	0.02 (0.00)	0.24 (0.01)	0.01	0.01 (0.00)	**0.96**
	Chess	0.66 (0.00)	0.84 (0.00)	0.01	0.70 (0.00)	0.01
	Connect-4	0.28 (0.00)	0.34 (0.00)	0.01	0.29 (0.00)	**0.19**
	Letter	0.36 (0.00)	0.96 (0.00)	0.01	0.26 (0.00)	**1.00**
	Shuttle	0.10 (0.01)	0.47 (0.24)	0.01	0.04 (0.01)	**1.00**

Table 4: Classification errors and empirical p-values obtained with Naive Bayes classifier for Test 1 and Test 2. The empirical p-values are calculated over 1000 randomized samples for small data sets and over 100 randomized samples for medium and large data sets. Classification on the original data is repeated ten times. In the table, the average values and standard deviations of the classification errors are given. Bold p-values correspond to nonsignificant results when the false discovery rate is restricted below 0.05 with Benjamini and Hochberg (1995) approach.

1-Nearest Neighbor

		Original	Test 1		Test 2	
	Data Set	Err. (Std)	Err. (Std)	p-val.	Err. (Std)	p-val.
Small	Audiology	0.26 (0.01)	0.86 (0.03)	0.001	0.32 (0.03)	**0.030**
	Autos	0.26 (0.01)	0.77 (0.03)	0.001	0.45 (0.03)	0.001
	Breast	0.31 (0.02)	0.41 (0.03)	0.007	0.32 (0.03)	**0.324**
	Glass	0.30 (0.01)	0.74 (0.04)	0.001	0.42 (0.03)	0.001
	Hepatitis	0.19 (0.01)	0.33 (0.04)	0.002	0.14 (0.03)	**0.970**
	Ionosphere	0.13 (0.00)	0.46 (0.03)	0.001	0.26 (0.01)	0.001
	Iris	0.05 (0.00)	0.66 (0.05)	0.001	0.02 (0.01)	**0.962**
	Lymph	0.18 (0.02)	0.53 (0.04)	0.001	0.20 (0.03)	**0.307**
	Promoters	0.19 (0.02)	0.50 (0.06)	0.001	0.26 (0.04)	**0.083**
	Segment	0.14 (0.01)	0.86 (0.03)	0.001	0.15 (0.02)	**0.266**
	Sonar	0.13 (0.01)	0.50 (0.04)	0.001	0.27 (0.03)	0.001
	Spect	0.24 (0.02)	0.32 (0.04)	0.011	0.18 (0.02)	**0.970**
	Tumor	0.66 (0.02)	0.88 (0.02)	0.001	0.62 (0.02)	**0.860**
	Votes	0.08 (0.01)	0.47 (0.03)	0.001	0.01 (0.00)	**1.000**
	Zoo	0.03 (0.01)	0.75 (0.05)	0.001	0.04 (0.02)	**0.333**
Medium	Abalone	0.80 (0.00)	0.90 (0.00)	0.01	0.68 (0.01)	**1.00**
	Anneal	0.05 (0.00)	0.40 (0.02)	0.01	0.08 (0.01)	0.01
	Balance	0.20 (0.01)	0.57 (0.02)	0.01	0.35 (0.02)	0.01
	Car	0.22 (0.01)	0.41 (0.05)	0.01	0.29 (0.01)	0.01
	German	0.28 (0.01)	0.42 (0.02)	0.01	0.33 (0.02)	0.01
	Mushroom	0.00 (0.00)	0.50 (0.01)	0.01	0.01 (0.00)	0.01
	Musk	0.04 (0.00)	0.26 (0.00)	0.01	0.53 (0.01)	0.01
	Pima	0.29 (0.00)	0.45 (0.02)	0.01	0.27 (0.02)	**0.88**
	Satellite	0.10 (0.00)	0.81 (0.01)	0.01	0.01 (0.00)	**1.00**
	Spam	0.09 (0.00)	0.48 (0.01)	0.01	0.09 (0.00)	**0.31**
	Splice	0.24 (0.01)	0.61 (0.01)	0.01	0.30 (0.01)	0.01
	Tic-tac-toe	0.21 (0.02)	0.44 (0.07)	0.01	0.38 (0.02)	0.01
	Yeast	0.47 (0.01)	0.78 (0.01)	0.01	0.52 (0.01)	0.01
Large	Adult	0.02 (0.00)	0.36 (0.00)	0.01	0.01 (0.00)	**1.00**
	Chess	0.48 (0.00)	0.90 (0.00)	0.01	0.80 (0.00)	0.01
	Connect-4	0.34 (0.00)	0.50 (0.00)	0.01	0.43 (0.00)	0.01
	Letter	0.06 (0.00)	0.96 (0.00)	0.01	0.46 (0.00)	0.01
	Shuttle	0.00 (0.00)	0.36 (0.00)	0.01	0.02 (0.00)	0.01

Table 5: Classification errors and empirical p-values obtained with 1-Nearest Neighbor classifier for Test 1 and Test 2. The empirical p-values are calculated over 1000 randomized samples for small data sets and over 100 randomized samples for medium and large data sets. Classification on the original data is repeated ten times. In the table, the average values and standard deviations of the classification errors are given. Bold p-values correspond to nonsignificant results when the false discovery rate is restricted below 0.05 with Benjamini and Hochberg (1995) approach.

Support Vector Machine

		Original	Test 1		Test 2	
	Data Set	Err. (Std)	Err. (Std)	p-val.	Err. (Std)	p-val.
Small	Audiology	0.20 (0.01)	0.83 (0.03)	0.001	0.20 (0.02)	**0.443**
	Autos	0.30 (0.02)	0.73 (0.04)	0.001	0.26 (0.03)	**0.873**
	Breast	0.30 (0.01)	0.31 (0.01)	**0.191**	0.25 (0.02)	**0.970**
	Glass	0.42 (0.01)	0.65 (0.03)	0.001	0.43 (0.02)	**0.363**
	Hepatitis	0.14 (0.01)	0.21 (0.00)	0.001	0.08 (0.02)	**0.999**
	Ionosphere	0.12 (0.01)	0.37 (0.01)	0.001	0.08 (0.01)	**0.995**
	Iris	0.04 (0.01)	0.67 (0.05)	0.001	0.02 (0.01)	**0.990**
	Lymph	0.14 (0.01)	0.51 (0.05)	0.001	0.12 (0.03)	**0.686**
	Promoters	0.09 (0.01)	0.50 (0.06)	0.001	0.10 (0.03)	**0.455**
	Segment	0.12 (0.01)	0.86 (0.03)	0.001	0.12 (0.01)	**0.529**
	Sonar	0.23 (0.02)	0.49 (0.04)	0.001	0.10 (0.02)	**0.999**
	Spect	0.17 (0.01)	0.21 (0.00)	0.001	0.08 (0.02)	**1.000**
	Tumor	0.53 (0.01)	0.77 (0.01)	0.001	0.53 (0.02)	**0.406**
	Votes	0.04 (0.00)	0.39 (0.01)	0.001	0.01 (0.00)	**1.000**
	Zoo	0.04 (0.00)	0.66 (0.04)	0.001	0.04 (0.01)	**0.666**
Medium	Abalone	0.75 (0.00)	0.84 (0.00)	0.01	0.57 (0.01)	**1.00**
	Anneal	0.15 (0.00)	0.24 (0.00)	0.01	0.14 (0.01)	**0.78**
	Balance	0.12 (0.01)	0.54 (0.03)	0.01	0.25 (0.01)	0.01
	Car	0.06 (0.00)	0.30 (0.00)	0.01	0.25 (0.01)	0.01
	German	0.25 (0.00)	0.30 (0.00)	0.01	0.22 (0.01)	**1.00**
	Mushroom	0.00 (0.00)	0.50 (0.01)	0.01	0.00 (0.00)	0.01
	Musk	0.05 (0.00)	0.15 (0.00)	0.01	0.01 (0.00)	**1.00**
	Pima	0.23 (0.00)	0.35 (0.00)	0.01	0.21 (0.01)	**1.00**
	Satellite	0.13 (0.00)	0.77 (0.00)	0.01	0.00 (0.00)	**1.00**
	Spam	0.10 (0.00)	0.39 (0.00)	0.01	0.04 (0.00)	**1.00**
	Splice	0.07 (0.00)	0.48 (0.00)	0.01	0.06 (0.01)	**0.99**
	Tic-tac-toe	0.02 (0.00)	0.37 (0.01)	0.01	0.30 (0.01)	0.01
	Yeast	0.43 (0.00)	0.69 (0.01)	0.01	0.42 (0.01)	**0.72**
Large	Adult	0.00 (0.00)	0.24 (0.00)	0.01	0.00 (0.00)	**1.00**
	Chess	0.66 (0.01)	0.85 (0.00)	0.01	0.72 (0.00)	0.01
	Connect-4	0.24 (0.00)	0.45 (0.07)	0.01	0.29 (0.00)	0.01
	Letter	0.19 (0.01)	0.96 (0.00)	0.01	0.32 (0.00)	0.01
	Shuttle	0.04 (0.01)	0.21 (0.00)	0.01	0.04 (0.00)	**0.45**

Table 6: Classification errors and empirical p-values for the Support Vector Machine classifier for Test 1 and Test 2. The empirical p-values are calculated over 1000 randomized samples for small data sets and over 100 randomized samples for medium and large data sets. Classification on the original data is repeated ten times. In the table, the average values and standard deviations of the classification errors are given. Bold p-values correspond to nonsignificant results when the false discovery rate is restricted below 0.05 with Benjamini and Hochberg (1995) approach.

pretty simple, or it is not used by the classification algorithm. That is, the classes differ from each other, from the point of view of the classifiers, mainly due to their different value distributions of the features and not due to some dependency between the features. Thus, in many data sets the class structure is explained by considering the features independently of each other.

The results with Naive Bayes classifier are in line with the analysis in Section 4.2. That is, practically all of the results are nonsignificant with Naive Bayes with Test 2 as it explicitly assumes independence of the features. However, there are three data sets where the results are regarded as significant with Test 2: Balance, Car and Chess. These three data sets seem to contain a good balance between the features that makes the Naive Bayes classifier to perform better on the original data than on the randomized data sets. That is, each instance contains usually at least one feature which makes the classification easy whereas in the randomized data sets there are instances that do not have separating values in any of the features. Thus, applying Test 2 to Naive Bayes classifier does not tell whether the classifier uses the interdependency between the features but whether the data are such that usually at least one feature in each instance has a clear separating value.

Compared to the other three classifiers, Naive Bayes is having both better and worse performance with all kind of data sets. Surprisingly, however, Naive Bayes is performing better also in a few such cases where the other classifiers are exploiting the feature dependency. For example, with data sets Splice and Yeast the Naive Bayes classifier has the best accuracy although the Decision Tree and 1-Nearest Neighbor classifiers are significant with Test 2. Thus if a classifier is using the feature dependency in the classification, it does not directly imply that some other classifier could not do better without using the dependency. In such case, however, it is likely that neither of the classifiers are optimal and we could obtain even better performance by combining the good properties of both the classifiers.

In the rest of this section, we will consider only the three other classifiers, namely Decision Tree, 1-Nearest Neighbor and Support Vector Machine classifiers. There is a clear difference between the small and large data sets with these classifiers. The results with Test 2 for small data sets are almost all nonsignificant whereas the results for large data sets are almost all significant. Only the Adult data set from large data sets seems to contain simple class structure. Actually, the Decision Tree and Support Vector Machine classifiers are able to classify correctly all the test samples on the original Adult data set as well as on the randomized versions of the Adult data set of Test 2. The results with the studied small UCI data sets are understandable, as many of them are known to contain fairly simple structure.

The results with the three classifiers are close to each other with all tests. Surprisingly, however, 1-Nearest Neighbor classifier has been able to use the interdependency between the features the most, that is, it contains the most of small, significant p-values with Test 2. However, other more complex classifiers could be able to find more data sets where the dependency between the features is useful in classification.

Let us now study the results with Test 2 in more detail. Consider the well-known Iris data set that contains measurements of three different species of iris flowers from four features: the length and the width of sepal and petal. It turns out that the classes are almost linearly separable given the length of petal or given the width of petal. Although there is a high positive linear correlation between the length and width of petal, it is not important for the classification result as both features can explain the classes by themselves.

Actually, observe that for the Iris data set with Test 2, the classification error on the randomized samples is even smaller than in the original data set. This phenomenon is explained by the

positive linear correlation between the length and the width of petal, which disappears after the randomizations, as seen in Figure 2 in Section 3.1. Randomizations eliminate most of the rows containing extreme values for both of the features inside the classes. Thus, the classifiers do not use the dependency between these two features, as their correlation does not help in classifying the Iris data. When this positive correlation is eliminated per classes, the separation between the classes increases, and therefore, the classification accuracy is improved.

For most of the data sets where the empirical p-values are very high for the null hypothesis of Test 2, there are either outliers inside the classes or positive correlation between some features that is not used in the classification as it does not help in separating the classes. For example, the data set Votes contains congressional "yes" and "no" voting records from republicans and democrats. There are few voting cases where the opinion of the voter clearly reveals the political views. However, there are some outliers, that is, people who have behaved more like democrats although they are republicans, or vice versa, that vanish after randomization. Nevertheless, these reasons do not remove the fact that the features independently classify the voting records.

Finally, we discuss the results for the Balance data set. With all classifiers the classification results of the Balance data set are significant under the null hypothesis of Test 2, that is, the classifiers have exploited the dependency between the features. The structure of the data supports this: The data contains four features of a balance scale: left-weight, left-distance, right-weight and right-distance. The scale is in balance if left-weight times left-distance equals right-weight times right-distance. There are three classes: the scale tips to the left, to the right, or is in balance. It is clear that the dependency between the features is necessary for correct classification result.

Note however, that understanding the structure inside the data sets where the classification results are regarded as significant under the null hypothesis of Test 2 requires more study, that is, we just know that the features do not explain the class structure independently. Analyzing the dependency structure of the features is then a further task. But as seen, the null hypothesis of Test 2 explains about half of the good classification results in the 33 data sets.

We conclude the experiments with a summary about the running times of the methods. We used MATLAB for producing the randomized data sets and Weka for performing the classification on a 2.2 GHz Opteron with 4 GB of main memory. The running times of producing one randomization of each data set and the running times of calculating the classification errors on the original data sets and on the randomized data sets are given in Table 7. The running times of producing the randomized data sets are negligible compared to the running times of calculating the classification errors of the data sets, that is, training and testing the classifiers. There is, however, a small difference between the running times of obtaining the classification errors on the original and the randomized data sets. Usually, the classification is a little bit faster on the original data set than on the randomized data sets. Furthermore, the classification on randomized data sets of Test 2 is usually faster than on randomized data sets of Test 1. The reason is that it is harder to teach a classifier on a randomized data set which has usually a weaker class structure than the original data set. Among the two randomization tests, Test 2 generally preserves the original class structure the most because it preserves some connection between the data and the class labels.

6. Conclusions

We have considered the problem of assessing the classifier performance with permutation tests in the statistical framework of hypothesis testing. We have described two different null hypotheses and

		Rand.		Decision Tree			Naive Bayes			1-Near. Neighbor			Supp. Vect. Mach.		
	Data Set	T1	T2	Or.	T1	T2	Or.	T1	T2	Or.	T1	T2	Or.	T1	T2
Small	Audiology	0.0	0.0	1.9	2.0	1.5	1.8	1.8	1.8	1.8	1.5	1.8	39	36	36
	Autos	0.0	0.0	0.5	0.5	0.5	0.5	0.4	0.4	0.5	0.4	0.4	3.2	2.2	2.3
	Breast	0.0	0.0	0.5	0.4	0.4	0.4	0.4	0.4	0.5	0.5	0.4	1.1	0.9	0.7
	Glass	0.0	0.0	0.4	0.4	0.3	0.3	0.3	0.2	0.3	0.2	0.2	2.4	2.3	2.3
	Hepatitis	0.0	0.0	0.4	0.3	0.3	0.3	0.3	0.2	0.3	0.3	0.3	0.5	0.4	0.4
	Ionosphere	0.0	0.0	1.2	1.0	0.9	0.8	0.7	0.7	0.9	0.8	0.9	1.2	1.3	1.0
	Iris	0.0	0.0	0.2	0.2	0.1	0.2	0.1	0.1	0.2	0.1	0.1	0.5	0.6	0.4
	Lymph	0.0	0.0	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	1.2	1.2	1.1
	Promoters	0.0	0.0	0.8	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6	1.0	0.9	0.8
	Segment	0.0	0.0	0.5	0.6	0.4	0.4	0.3	0.3	0.4	0.3	0.4	3.6	2.4	2.3
	Sonar	0.0	0.0	1.2	0.9	1.1	0.9	0.7	0.7	0.9	0.9	0.9	1.1	1.1	0.9
	Spect	0.0	0.0	0.7	0.6	0.6	0.6	0.8	0.6	0.8	0.6	0.6	0.8	0.8	0.7
	Tumor	0.0	0.0	0.9	0.8	0.8	0.8	0.7	0.6	0.8	0.8	0.8	30	21	26
	Votes	0.0	0.0	1.0	0.8	0.7	0.9	0.9	0.7	1.0	0.9	0.8	1.2	1.1	1.1
	Zoo	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	3.1	3.1	2.6
Medium	Abalone	0.0	0.0	7.0	8.9	6.7	3.1	3.0	3.0	7.6	7.8	7.8	54	78	60
	Anneal	0.0	0.0	2.9	2.9	2.9	2.3	2.3	2.3	3.1	3.1	3.2	4.7	9.3	4.6
	Balance	0.0	0.0	0.5	0.5	0.5	0.4	0.3	0.4	0.5	0.5	0.4	0.9	1.0	0.8
	Car	0.0	0.0	1.4	1.5	1.4	1.4	1.4	1.3	1.8	1.8	1.7	5.1	8.0	7.7
	German	0.0	0.0	1.8	1.5	1.5	1.6	1.2	1.5	2.0	2.1	2.1	9.4	6.3	9.8
	Mushroom	0.0	0.0	21	24	21	20	21	20	68	70	70	60	1197	26
	Musk	0.0	0.2	130	110	176	55	63	80	230	309	318	502	5816	86
	Pima	0.0	0.0	0.8	0.7	0.9	0.7	0.7	0.5	0.8	0.7	0.8	0.9	0.8	0.9
	Satellite	0.0	0.0	26	128	21	13	14	14	59	62	81	19	157	15
	Spam	0.0	0.1	32	16	23	14	14	14	39	56	56	21	38	17
	Splice	0.0	0.0	17	17	16	15	16	15	36	28	28	87	1922	95
	Tic-tac-toe	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.1	1.1	1.2	3.1	3.5	6.6
	Yeast	0.0	0.0	1.7	2.2	1.6	1.3	1.0	1.3	1.4	1.4	1.7	5.2	5.3	4.9
Large	Adult	0.0	0.4	55	60	58	56	62	67	315	325	304	77	134	63
	Chess	0.0	0.1	71	57	54	45	54	59	131	105	111	93	89	102
	Connect-4	0.0	0.9	379	380	292	387	391	291	1161	1297	1285	975	558	1066
	Letter	0.0	0.1	37	45	42	32	45	34	95	103	99	43	50	45
	Shuttle	0.0	0.2	176	139	142	141	143	138	339	419	319	149	170	140

Table 7: Average running times in seconds for obtaining one randomization version of each data set for Test 1 (T1) and Test 2 (T2), as well as running times for obtaining one classification error for the four studied classifiers on each original data set (Or.) and on each randomized version of each data set (T1, T2). The running times are the average values over all the samples produced. Note that the classification procedures for small, medium and large data sets differ from each other.

shown how samples can be produced from the corresponding null models by simple permutation methods. Each test provides an empirical p-value for the classifier performance; each p-value depends on the original data (whether it contains the type of structure tested) and the classifier (whether it is able to use the structure). The two null hypotheses can be summarized as follows: (1) the data and the class labels are independent; and (2) the features are mutually independent given the class label.

Each test evaluates whether a certain structure (label–class dependency or dependency between features inside a class) is present in the data, and whether the classifier can use such structure for obtaining good results. If the original data really contains the structure being tested, then a significant classifier should use such information and thus obtain a low p-value. If the classifier is not significant then it will not notice the structure from the original data and thus, get a high p-value. On the other hand, if the original data does not contain any structure at all, then all p-values should be very high.

We have performed extensive experiments both on synthetic and real data. Experiments showed that the traditional permutation test (i.e., data and class labels are independent) is not useful in studying real data sets as it produces a small p-value even if there is only a weak class structure present. Compared to this, the new test proposed, that is, permuting the features inside a class, was able to evaluate the underlying reasons for the classifier performance on the real data sets. Surprisingly, however, in about half of the studied real data sets the class structure looks fairly simple; the dependency between the features is not used in classifying the data with the four tested classifiers. In such cases, there might be no reason to use the chosen classifier. That is, either the same or even better performance could be obtained by using some simpler methods, or the classification performance could be improved further by taking some useful unused feature dependency into account by changing the classification algorithm.

Interpreting the descriptive information provided by Test 2 needs care. If the classifier is significant with Test 2, then the data really contains a feature dependency that the classifier is exploiting. However, if the classifier is not significant with Test 2, that is, we obtain a high p-value, there are three different possibilities: (1) there are no dependencies between the features in the data; (2) there are some dependencies between the features in the data but they do not increase the class separation; or (3) there are useful dependencies between the features in the data that increase the class separation but the chosen classifier is not able to exploit them. In the third case, we would like to find such a classifier that could use the feature dependency to improve the classification performance. However, in general, when a high p-value is obtained with Test 2, we cannot know which of these applies to the data and to the chosen classifier. Thus the best we can do is to continue the search for a better classifier by assuming that any of them could be true. That is, we try more complex classifiers that could use the possible existing feature dependency, as well as simpler classifiers that could perform better if no feature dependency exists. Nevertheless, the answer provided by Test 2 is definite, that is, it tells whether the chosen classifier uses feature dependency to improve the classification performance.

Future work should explore the use of Test 2 for selecting the best discriminant features for classifiers, in similar fashion as Test 1 has been used for decision trees and other biological applications (Frank, 2000; Frank and Witten, 1998; Maglietta et al., 2007). Also, it would be useful to extend the setting to unsupervised learning, such as clustering. In addition, more study is needed for exploiting the descriptive information provided by Test 2. Specifically, how should we proceed to improve and study the classification performance when a high p-value is obtained with Test 2?

Acknowledgments

The authors would like to thank the editor and the anonymous reviewers for their careful reading and for the useful comments that improved substantially this manuscript. Thanks also to Antti Ukkonen and Kai Puolamäki for discussions on the previous version of this paper.

References

Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

Alain Berlinet, Gérard Biau, and Laurent Rouvière. Functional supervised classification with wavelets. *Annales de l'ISUP*, 52:61–80, 2008.

Julian Besag and Peter Clifford. Sequential Monte Carlo p-values. *Biometrika*, 78(2):301–304, 1991.

Howard D. Bondell and Brian J. Reich. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with OSCAR. *Biometrics*, 64:115–123, 2008.

Ulisses Braga-Neto and Edward R. Dougherty. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374–380, 2004.

George Casella and Roger L. Berger. *Statistical Inference*. Duxbury Resource Center, 2001.

Yuguo Chen, Persi Diaconis, Susan P. Holmes, and Jun S. Liu. Sequential Monte Carlo methods for statistical analysis of tables. *Journal of the American Statistical Association*, 100(469):109–120, 2005.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

Michael P. Fay, Hyune-Ju Kim, and Mark Hachey. On using truncated sequential probability ratio test boundaries for Monte Carlo implementation of hypothesis tests. *Journal of Computational and Graphical Statistics*, 16(4):946–967, December 2007.

Eibe Frank. *Pruning Decision Trees and Lists*. PhD thesis, University of Waikato, 2000.

Eibe Frank and Ian H. Witten. Using a permutation test for attribute selection in decision trees. In *International Conference on Machine Learning*, pages 152–160, 1998.

Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas. Assessing data mining results via swap randomization. *ACM Trans. Knowl. Discov. Data*, 1(3), 2007.

Polina Golland and Bruce Fischl. Permutation tests for classification: Towards statistical significance in image-based studies. In *International Conference on Information Processing and Medical Imaging*, pages 330–341, 2003.

Polina Golland, Feng Liang, Sayan Mukherjee, and Dmitry Panchenko. Permutation tests for classification. In *Annual Conference on Learning Theory*, pages 501–515, 2005.

Phillip I. Good. *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses; Springer series in statistics.*, volume 2nd. Springer, 2000.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

Tailen Hsing, Sanju Attoor, and Edward R. Dougherty. Relation between permutation-test p values and classifier error estimates. *Mach. Learn.*, 52(1-2):11–30, 2003.

Anders Isaksson, Mikael Wallman, Hanna Göransson, and Mats G. Gustafsson. Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recogn. Lett.*, 29(14): 1960–1965, 2008.

David Jensen. *Induction with Randomization Testing: Decision-Oriented Analysis of Large Data Sets*. PhD thesis, Washington University, St. Louis, Missouri, USA, 1992.

Rosalia Maglietta, Annarita D'Addabbo, Ada Piepoli, Francesco Perri, Sabino Liuni, Graziano Pesole, and Nicola Ancona. Selection of relevant genes in cancer diagnosis based on their prediction accuracy. *Artif. Intell. Med.*, 40(1):29–44, 2007.

Annette M. Molinaro, Richard Simon, and Ruth M. Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307, 2005.

Markus Ojala and Gemma C. Garriga. Permutation tests for studying classifier performance. In *Proceedings of the 9th IEEE International Conference on Data Mining*, pages 908–913, 2009.

Markus Ojala, Niko Vuokko, Aleksi Kallio, Niina Haiminen, and Heikki Mannila. Randomization methods for assessing data analysis results on real-valued matrices. *Statistical Analysis and Data Mining*, 2(4):209–230, 2009.

Fabrice Rossi and Nathalie Villa. Support vector machine for functional data classification. *Neurocomputing*, 69(7-9):730–742, 2006.

Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.

Journal of Machine Learning Research 11 (2010) 1865-1881 Submitted 2/08; Revised 2/10; Published 6/10

Sparse Spectrum Gaussian Process Regression

Miguel Lázaro-Gredilla MIGUEL@TSC.UC3M.ES
Departamento de Teoría de la Señal y Comunicaciones
Universidad Carlos III de Madrid
28911 Leganés, Madrid, Spain

Joaquin Quiñonero-Candela JOAQUINC@MICROSOFT.COM
Microsoft Research Ltd.
7 J J Thomson Avenue
Cambridge CB3 0FB, UK

Carl Edward Rasmussen* CER54@CAM.AC.UK
Deparment of Engineering
University of Cambridge, Trumpington st.
Cambridge CB2 1PZ, UK

Aníbal R. Figueiras-Vidal ARFV@TSC.UC3M.ES
Departamento de Teoría de la Señal y Comunicaciones
Universidad Carlos III de Madrid
28911 Leganés, Madrid, Spain

Editor: Tommi Jaakkola

Abstract

We present a new sparse Gaussian Process (GP) model for regression. The key novel idea is to sparsify the *spectral representation* of the GP. This leads to a simple, practical algorithm for regression tasks. We compare the achievable trade-offs between predictive accuracy and computational requirements, and show that these are typically superior to existing state-of-the-art sparse approximations. We discuss both the weight space and function space representations, and note that the new construction implies priors over functions which are always stationary, and can approximate any covariance function in this class.

Keywords: Gaussian process, probabilistic regression, sparse approximation, power spectrum, computational efficiency

1. Introduction

One of the main practical limitations of Gaussian processes (GPs) for machine learning (Rasmussen and Williams, 2006) is that in a direct implementation the computational and memory requirements scale as $O(n^2)$ and $O(n^3)$, respectively. In practice this limits the applicability of exact GP implementations to data sets where the number of training samples n does not exceed a few thousand.

A number of computationally efficient approximations to GPs have been proposed, which reduce storage requirements to $O(nm)$ and the number of computations to $O(nm^2)$, where m is much smaller than n. One family of approximations, reviewed in Quiñonero-Candela and Rasmussen (2005), is based on assumptions of conditional independence given a reduced set of *m inducing*

*. Also at Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany .

inputs. Examples of such models are those proposed in Seeger et al. (2003), Smola and Bartlett (2001), Tresp (2000), Williams and Seeger (2001) and Csató and Opper (2002), as well as the Fully Independent Training Conditional (FITC) model, introduced as Sparse Pseudo-input GP (SPGP) by Snelson and Ghahramani (2006). Walder et al. (2008) introduced the Sparse Multiscale GP (SMGP), a modification of FITC that allows each basis function to have its own set of length-scales.[1] This additional flexibility typically yields some performance improvement over FITC, but it also requires learning twice as many parameters. SMGP can alternatively be derived within the unifying framework of Quiñonero-Candela and Rasmussen (2005) if we allow the inducing inputs to lie in a transformed domain, as shown in Lázaro-Gredilla and Figueiras-Vidal (2010).

Another family of approximations is based on approximate matrix-vector-multiplications (MVMs), where m is for example a reduced number of conjugate gradient steps to solve a system of linear equations. Some of these methods have been briefly reviewed in Quiñonero-Candela et al. (2007). Local mixtures of GP have been used by Urtasun and Darrell (2008) for efficient modelling of human poses.

In this paper we introduce a stationary trigonometric Bayesian model for regression that retains the computational efficiency of the aforementioned approaches, while improving performance. The model consists of a linear combination of trigonometric functions where both weights and phases are integrated out. All hyperparameters of the model (frequencies and amplitudes) are learned jointly by maximizing the marginal likelihood. This model is a stationary sparse GP that can approximate any desired stationary full GP. Sparse trigonometric expansions have been proposed in several contexts, for example, Lázaro-Gredilla et al. (2007) and Rahimi and Recht (2008), as discussed further in Section 4.3.

FITC, SMGP, and the model introduced in this paper focus on predictive accuracy at low computational cost, rather than on faithfully converging towards the full GP as the number of basis functions grows. Performance-wise, FITC and the more recent SMGP can be regarded as the current state-of-the-art sparse GP approximations, so we will use them as benchmarks in the performance comparisons.

In Section 2 we give a brief review of GP regression. In Section 3 we introduce the trigonometric Bayesian model, and in Section 4 we present the Sparse Spectrum Gaussian Process (SSGP) algorithm. Section 5 contains a comparative performance evaluation on several data sets.

2. Gaussian Process Regression

Regression is often formulated as the task of predicting the scalar output y_* associated to the D-dimensional input $\mathbf{x}_*$, given a training data set $\mathcal{D} \equiv \{\mathbf{x}_j, y_j | j = 1, \dots n\}$ of n input-output pairs. A common approach is to assume that the outputs have been generated by an unknown latent function $f(\mathbf{x})$ and independently corrupted by additive Gaussian noise of constant variance σ_n^2:

$$y_j \;=\; f(\mathbf{x}_j) + \varepsilon_j, \qquad \varepsilon_j \;\sim\; \mathcal{N}(0,\, \sigma_n^2)\,.$$

The regression task boils down to making inference about $f(\mathbf{x})$. Gaussian process (GP) regression is a probabilistic, non-parametric Bayesian approach. A Gaussian process prior distribution on $f(\mathbf{x})$ allows us to encode assumptions about the smoothness (or other properties) of the latent function

1. Note that SMGP only extends FITC in the specific case of the anisotropic squared exponential covariance function, whereas FITC can be applied to any covariance function.

(Rasmussen and Williams, 2006). For any set of inputs $\{\mathbf{x}_i\}_{i=1}^n$ the corresponding vector of function evaluations $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^\top$ has a joint Gaussian distribution:

$$p(\mathbf{f}|\{\mathbf{x}_i\}_{i=1}^n) \;=\; \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K_{ff}}).$$

This paper follows the common practice of setting the mean of the process to zero.[2] The properties of the GP prior over functions are governed by the covariance function

$$\left[\mathbf{K_{ff}}\right]_{ij} \;=\; k(\mathbf{x}_i, \mathbf{x}_j) \;=\; \mathbb{E}[f(\mathbf{x}_i) f(\mathbf{x}_j)], \tag{1}$$

which determines how the similarity between a pair of function values varies as a function of the corresponding pair of inputs. A covariance function is *stationary* if it only depends on the difference between its inputs

$$k(\mathbf{x}_i, \mathbf{x}_j) \;=\; k(\mathbf{x}_i - \mathbf{x}_j) \;=\; k(\boldsymbol{\tau}).$$

The elegance of the GP framework is that the properties of the function are conveniently expressed directly in terms of the covariance function, rather than implicitly via basis functions.

To obtain the predictive distribution $p(y_*|\mathbf{x}_*, \mathcal{D})$ it is useful to express the model in matrix notation by stacking the targets y_j in vector $\mathbf{y} = [y_1, \ldots, y_n]^\top$ and writing the joint distribution of training and test targets:

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \;\sim\; \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K_{ff}} + \sigma_n^2 \mathbf{I}_n & \mathbf{k_{f*}} \\ \mathbf{k_{f*}}^\top & k_{**} + \sigma_n^2 \end{bmatrix}\right),$$

where $\mathbf{k_{f*}}$ is the vector of covariances between $f(\mathbf{x}_*)$ and the training latent function values, and k_{**} is the prior variance of $f(\mathbf{x}_*)$. $\mathbf{I}_n$ is the $n \times n$ identity. The predictive distribution is obtained by conditioning on the observed training outputs:

$$p(y_*|\mathbf{x}_*, \mathcal{D}) \;=\; \mathcal{N}(\mu_*, \sigma_*^2), \;\; \text{where} \;\; \begin{cases} \mu_* = \mathbf{k}_{*\mathbf{f}}(\mathbf{K_{ff}} + \sigma_n^2 \mathbf{I}_n)^{-1}\mathbf{y} \\ \sigma_*^2 = \sigma_n^2 + k_{**} - \mathbf{k}_{*\mathbf{f}}(\mathbf{K_{ff}} + \sigma_n^2 \mathbf{I}_n)^{-1}\mathbf{k_{f*}}. \end{cases} \tag{2}$$

The covariance function is parameterized by *hyperparameters*. Consider for example the stationary anisotropic squared exponential covariance function

$$k_{\text{ARD}}(\boldsymbol{\tau}) \;=\; \sigma_0^2 \exp(-\tfrac{1}{2}\boldsymbol{\tau}^\top \Lambda^{-1} \boldsymbol{\tau}), \quad \text{where} \quad \Lambda \;=\; \text{diag}([\ell_1^2, \ell_2^2, \ \ldots \ \ell_D^2]). \tag{3}$$

The hyperparameters are the prior variance σ_0^2 and the lengthscales $\{\ell_d\}$ that determine how rapidly the covariance decays with the distance between inputs. This covariance function is also known as the ARD (Automatic Relevance Determination) squared exponential, because it can effectively prune input dimensions by growing the corresponding lengthscales.

It is convenient to denote all hyperparameters including the noise variance by θ. These can be learned by maximizing the evidence, or log marginal likelihood:

$$\log p(\mathbf{y}|\theta) \;=\; -\frac{n}{2}\log(2\pi) - \frac{1}{2}|\mathbf{K_{ff}} + \sigma_n^2 \mathbf{I}_n| - \frac{1}{2}\mathbf{y}^\top \left(\mathbf{K_{ff}} + \sigma_n^2 \mathbf{I}_n\right)^{-1} \mathbf{y}. \tag{4}$$

Provided there exist analytic forms for the gradients of the covariance function with respect to the hyperparameters, the evidence can be maximized by using a gradient-based search. Unfortunately, computing the evidence and the gradients requires the inversion of the covariance matrix $\mathbf{K_{ff}} + \sigma_n^2 \mathbf{I}_n$ at a cost of $O(n^3)$ operations, which is prohibitive for large data sets.

2. The extension to GPs with general mean functions is straightforward.

3. Trigonometric Bayesian Regression

In this section we present a Bayesian linear regression model with trigonometric basis functions, and related it to a full GP in the next section. Consider the model

$$f(\mathbf{x}) \;=\; \sum_{r=1}^{m} a_r \cos(2\pi \mathbf{s}_r^\top \mathbf{x}) + b_r \sin(2\pi \mathbf{s}_r^\top \mathbf{x}) \,, \tag{5}$$

where each of the m pairs of basis functions is parametrized by a D-dimensional vector $\mathbf{s}_r$ of spectral frequencies. Note that each *pair* of basis functions share frequencies, but each have independent amplitude parameters, a_r and b_r. We treat the frequencies as deterministic parameters and the amplitudes in a Bayesian way. The priors are independent Gaussian

$$a_r \;\sim\; \mathcal{N}\big(0, \frac{\sigma_0^2}{m}\big) \,, \quad b_r \;\sim\; \mathcal{N}\big(0, \frac{\sigma_0^2}{m}\big) \,,$$

where the variances are scaled down linearly by the number of basis functions. Under the prior, the distribution over functions from Equation (5) is Gaussian with mean function zero and covariance function (from Equation (1))

$$k(\mathbf{x}_i, \mathbf{x}_j) \;=\; \frac{\sigma_0^2}{m} \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \;=\; \frac{\sigma_0^2}{m} \sum_{r=1}^{m} \cos\left(2\pi \mathbf{s}_r^\top (\mathbf{x}_i - \mathbf{x}_j)\right) \,, \tag{6}$$

where we define the column vector of length $2m$ containing the evaluation of the m pairs of trigonometric functions at $\mathbf{x}$

$$\phi(\mathbf{x}) \;=\; \big[\cos(2\pi \mathbf{s}_1^\top \mathbf{x}) \;\sin(2\pi \mathbf{s}_1^\top \mathbf{x}) \;\ldots\; \cos(2\pi \mathbf{s}_m^\top \mathbf{x}) \quad \sin(2\pi \mathbf{s}_m^\top \mathbf{x})\big]^\top .$$

Sparse linear models generally induce priors over functions whose variance depends on the input. In contrast, the covariance function in Equation (6) is stationary, that is, the prior variance is independent of the input and equal to σ_0^2. This is due to the particular nature of the trigonometric basis functions and implies that the predictive variances cannot be "healed", as proposed in Rasmussen and Quiñonero-Candela (2005) for the case of the Relevance Vector Machine.

The predictions and marginal likelihood can be evaluated using Equations (2) and (4), although direct evaluation is computationally inefficient when $2m < n$. For the predictive distribution we use the more efficient

$$\mathbb{E}[y_*] \;=\; \phi(\mathbf{x}_*)^\top \mathbf{A}^{-1} \mathbf{\Phi}_\mathbf{f} \mathbf{y}, \quad \mathbb{V}[y_*] \;=\; \sigma_n^2 + \sigma_n^2 \phi(\mathbf{x}_*)^\top \mathbf{A}^{-1} \phi(\mathbf{x}_*), \tag{7}$$

where we have defined the $2m$ by n design matrix $\mathbf{\Phi}_\mathbf{f} = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)]$ and $\mathbf{A} = \mathbf{\Phi}_\mathbf{f} \mathbf{\Phi}_\mathbf{f}^\top + \frac{m\sigma_n^2}{\sigma_0^2} \mathbf{I}_{2m}$. Similarly, for the log marginal likelihood

$$\log p(\mathbf{y}|\theta) \;=\; -\big[\mathbf{y}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{\Phi}_\mathbf{f}^\top \mathbf{A}^{-1} \mathbf{\Phi}_\mathbf{f} \mathbf{y}\big]/(2\sigma_n^2) - \frac{1}{2} \log |\mathbf{A}| + m \log \frac{m\sigma_n^2}{\sigma_0^2} - \frac{n}{2} \log 2\pi\sigma_n^2 \,. \tag{8}$$

A stable and efficient implementation uses Cholesky decompositions, Appendix A. Both the predictive distribution and the marginal likelihood can be computed in $O(nm^2)$. The predictive mean and variance at an additional test point can be computed in $O(m)$ and $O(m^2)$ respectively. The storage costs are also reduced, since we no longer store the full covariance matrix (of size $n \times n$), but only the design matrix (of size $n \times 2m$).

3.1 Periodicity

One might be tempted to assume that this model would only be useful for modeling periodic functions, since strictly speaking a linear combination of periodic signals is itself periodic. However, if the individual frequencies are not all multiples of a common base frequency, then the period of the resulting signal will be very long, typically exceeding the range of the inputs by orders of magnitude. Thus, the model based on trigonometric basis functions has practical use for modeling non-periodic functions. The same principle is used (interchanging input and frequency domains) in uneven sampling to space apart frequency replicas an avoid aliasing, see for instance Bretthorst (2000). As our experimental results suggest, the model provides satisfactory predictive variances.

3.2 Representation

An alternative and equivalent representation of the model in Equation (5), which only uses half the number of trigonometric basis functions is possible, by writing the linear combination of a sine and a cosine as a cosine with an amplitude and a phase. Although these two representations are equivalent, *inference* based on them differs. Whereas we have been able to integrate out the amplitudes to arrive at the GP in Equation (6), this would not be possible analytically using the more parsimonious representation.

Optimization instead of marginalization of the phases has two important consequences. Firstly, we lose the property of stationarity of the prior over functions. Secondly we may expect that the model becomes more prone to overfitting. When considering the contribution from a basis function (pair) with a specific frequency, the optimization based scheme could fit arbitrarily the phase, whereas the integration based inference is constrained to use a flat prior over phases. In Section 5.3 we empirically verify that the computation vs accuracy tradeoff typically favors the less compact representation.

4. The Sparse Spectrum Gaussian Process

In the previous section we presented an explicit basis function regression model, but we did not discuss how to select the frequencies defining the basis functions. In this section, we present a sparse GP approximation view of this model, which shows how it can be understood as a computationally efficient approximation to any GP with stationary covariance function. In the next section we present experimental results showing that dramatic improvements over other state-of-the-art sparse GP regression algorithms are possible.

We will now take a generic GP with stationary covariance function and sparsify its power spectral density to obtain a sparse GP that approximates the full GP. The power spectral density (or power spectrum) $S(\mathbf{s})$ of a stationary random process expresses how the power is distributed over the frequency domain. For a stationary GP, the power is equal to the prior variance $k(\mathbf{x},\mathbf{x}) = k(\mathbf{0}) = \sigma_0^2$. The frequency vector $\mathbf{s}$ has the same length D as the input vector $\mathbf{x}$. The d-th element of $\mathbf{s}$ can be interpreted as the frequency associated to the d-th input dimension. The Wiener-Khintchine theorem (see for example Carlson, 1986, p. 162) states that the power spectrum and the autocorrelation of the random process constitute a Fourier pair. In our case, given that $f(\cdot)$ is drawn from a stationary Gaussian process, the autocorrelation function is equal to the stationary covariance function, and we have:

$$k(\boldsymbol{\tau}) = \int_{\mathbb{R}^D} e^{2\pi i \mathbf{s}^\top \boldsymbol{\tau}} S(\mathbf{s}) d\mathbf{s}, \qquad S(\mathbf{s}) = \int_{\mathbb{R}^D} e^{-2\pi i \mathbf{s}^\top \boldsymbol{\tau}} k(\boldsymbol{\tau}) d\boldsymbol{\tau}. \tag{9}$$

We thus see that there are two equivalent representations for a stationary Gaussian process: the traditional one in terms of the covariance function in the (input) space domain, and a perhaps less usual one as the power spectrum in the frequency domain.

Bochner's theorem (Stein, 1999, p. 24) states that any stationary covariance function $k(\boldsymbol{\tau})$ can be represented as the Fourier transform of a positive finite measure. This means that the power spectrum in (9) is a positive finite measure, and in particular that it is *proportional* to a probability measure, $S(\mathbf{s}) \propto p_S(\mathbf{s})$. The proportionality constant can be directly obtained by evaluating the covariance function in (9) at $\boldsymbol{\tau} = \mathbf{0}$. We obtain the relation:

$$S(\mathbf{s}) \;=\; k(\mathbf{0})\, p_S(\mathbf{s}) \;=\; \sigma_0^2\, p_S(\mathbf{s}). \tag{10}$$

We can use the fact that $S(\mathbf{s})$ is proportional to a multivariate probability density in $\mathbf{s}$ to rewrite the covariance function in (9) as an expectation:

$$\begin{aligned} k(\mathbf{x}_i, \mathbf{x}_j) \;&=\; k(\boldsymbol{\tau}) = \int_{\mathbb{R}^D} e^{2\pi i \mathbf{s}^\top (\mathbf{x}_i - \mathbf{x}_j)} S(\mathbf{s})\, \mathrm{d}\mathbf{s} \;=\; \sigma_0^2 \int_{\mathbb{R}^D} e^{2\pi i \mathbf{s}^\top \mathbf{x}_i} \left(e^{2\pi i \mathbf{s}^\top \mathbf{x}_j}\right)^* p_S(\mathbf{s}) d\mathbf{s} \\ &=\; \sigma_0^2 \mathbb{E}_{p_S}\left[e^{2\pi i \mathbf{s}^\top \mathbf{x}_i} \left(e^{2\pi i \mathbf{s}^\top \mathbf{x}_j}\right)^* \right], \end{aligned} \tag{11}$$

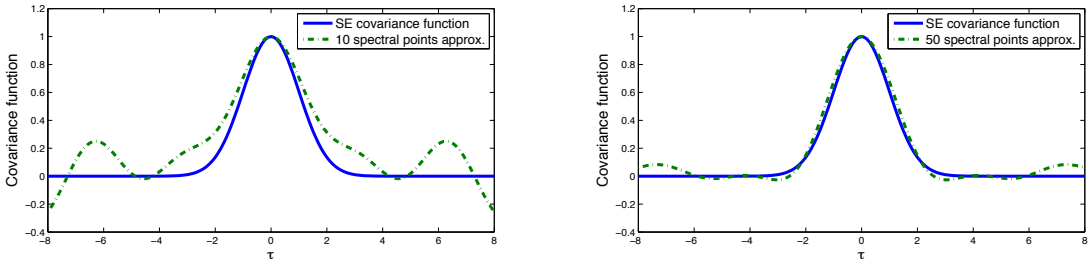
where $\mathbb{E}_{p_S}$ denotes expectation wrt. $p_S(\mathbf{s})$ and superscript asterisk[3] denotes complex conjugation. This last expression is an exact expansion of the covariance function as the expectation of a product of complex exponentials with respect to a particular distribution over their frequencies. This integral can be approximated by simple Monte Carlo by taking an average of a few samples corresponding to a finite set of frequencies, which we call *spectral points*.

Since the power spectrum is symmetric around zero, a valid Monte Carlo procedure is to sample frequencies always as a pair $\{\mathbf{s}_r, -\mathbf{s}_r\}$. This has the advantage of preserving the property of the exact expansion, Equation (11) that the imaginary terms cancel:

$$\begin{aligned} k(\mathbf{x}_i, \mathbf{x}_j) \;&\simeq\; \frac{\sigma_0^2}{2m} \sum_{r=1}^{m} \left[e^{2\pi i \mathbf{s}_r^\top \mathbf{x}_i} \left(e^{2\pi i \mathbf{s}_r^\top \mathbf{x}_j}\right)^* + \left(e^{2\pi i \mathbf{s}_r^\top \mathbf{x}_i}\right)^* e^{2\pi i \mathbf{s}_r^\top \mathbf{x}_j} \right] \\ &=\; \frac{\sigma_0^2}{m} \mathrm{Re}\Big[\sum_{r=1}^{m} e^{2\pi i \mathbf{s}_r^\top \mathbf{x}_i} \left(e^{2\pi i \mathbf{s}_r^\top \mathbf{x}_j}\right)^* \Big] \;=\; \frac{\sigma_0^2}{m} \sum_{r=1}^{m} \cos\left(2\pi \mathbf{s}_r^\top (\mathbf{x}_i - \mathbf{x}_j)\right), \end{aligned}$$

where $\mathbf{s}_r$ is drawn from $p_S(\mathbf{s})$ and $\mathrm{Re}[\cdot]$ denotes the real part of a complex number. Notice, that we have recovered exactly the expression for the covariance function induced by the trigonometric basis functions model, Equation (6). Further, we have given an interpretation of the frequencies as spectral Monte Carlo samples, approximating *any* stationary covariance function. This is a more general result than that of (MacKay, 2003, Ch. 45), which only applies to Gaussian covariances. The approximation is equivalent to replacing the original spectrum $S(\mathbf{s})$ by a set of Dirac deltas of amplitude σ_0^2 distributed according to $p_S(\mathbf{s})$. Thus, we "sparsify" the spectrum of the GP.

This convergence result can also be stated as follows: A stationary GP can be seen as a neural network with infinitely many hidden units and trigonometric activations if independent priors following Gaussian and $p_S(\mathbf{s})$ distributions are placed on the output and input weights, respectively. This is analogous to the result of Williams (1997) for the non-stationary multilayer perceptron covariance function.



(a) (b)

Figure 1: Squared exponential covariance function and its approximation with (a) 10 and (b) 50 random spectral points respectively.

4.1 Example: The Squared Exponential Covariance Function

The probability density associated to the squared exponential covariance function of Equation (3) can be obtained from the Fourier transform

$$p_S^{\mathrm{ARD}}(\mathbf{s}) = \frac{1}{k_{\mathrm{ARD}}(\mathbf{0})}\int_{\mathbb{R}^D} e^{-2\pi i \mathbf{s}^\top \boldsymbol{\tau}} k_{\mathrm{ARD}}(\boldsymbol{\tau})\, d\boldsymbol{\tau} = \sqrt{|2\pi\Lambda|}\exp(-2\pi^2 \mathbf{s}^\top \Lambda \mathbf{s}), \tag{12}$$

which also has the form of a multivariate Gaussian distribution. For illustration purposes, we compare the exact squared exponential covariance function with its spectral approximation in Figure 1, where the spectral points are sampled from Equation (12). As expected, the quality of the approximation improves with the number of samples.

4.2 The SSGP Algorithm

One of the main goals of sparse approximations is to reduce the computational burden while retaining as much predictive accuracy as possible. Sampling from the spectral density constitutes a way of building a sparse approximation. However, we may suspect that we can obtain much sparser models if the spectral frequencies are learned by optimizing the marginal likelihood, an idea which we pursue in the following.

The algorithm we propose uses conjugate gradients to optimize the marginal likelihood (8) with respect to the spectral points $\{\mathbf{s}_r\}$ and the hyperparameters σ_0^2, σ_n^2, and $\{\ell_1,\ \ell_2,\ \ldots\ \ell_D\}$. Optimizing with respect to the lengthscales in addition to the spectral points is effectively an overparametrization, but in our experience this redundancy proves helpful in avoiding undesired local minima. As is usual with this kind of optimization, the problem is non-convex and we cannot expect to find the global optimum. The goal of the optimization is to find a reasonable local optimum.

In detail, model selection for the SSGP algorithm consists in:

1. Initialize $\{\ell_d\}$, σ_0^2, and σ_n^2 to some sensible values. We use one half of the ranges of the input dimensions, the variance of $\{y_j\}$ and $\sigma_0^2/4$, respectively.

2. Initialize the $\{\mathbf{s}_r\}$ by sampling from (10).

3. The superscript asterisk denotes complex conjugate and the subscript asterisk indicates test quantity.

3. Jointly optimize the marginal likelihood wrt. spectral points and hyperparameters.

The computational cost of training the SSGP algorithm is $O(nm^2)$ per conjugate gradient step. At prediction time, the cost is $O(m)$ for the predictive mean and $O(m^2)$ for the predictive variance per test point. These computational costs are of the same order as those of the majority of the sparse GP approximations that have recently been proposed (see Quiñonero-Candela and Rasmussen, 2005, for a review).

Learning the spectral frequencies by optimization departs from the original motivation of approximating a full GP. The optimization stage poses a risk of overfitting, which we assess in the experimental section that follows. However, the additional flexibility can potentially improve performance since it allows learning a covariance function suitable to the problem at hand.

4.3 Related Algorithms

Finite decompositions in terms of harmonic basis functions, such as Fourier series, are a classic idea. In the context of kernel machines recent work include Lázaro-Gredilla et al. (2007) for GPs and Rahimi and Recht (2008) for Support Vector Machines (SVMs). As we show in the experimental section, the details of the implementation turn out to have a critical impact on the performance of the algorithms. The SVM based approach uses projections onto a random set of harmonic functions, whereas the approach used in this paper uses the evidence framework to carefully craft an optimized sparse harmonic representation. As is revealed in the experimental section, optimization of the frequencies, amplitudes and noise offers dramatic performance improvements for comparable sparseness.

5. Experiments

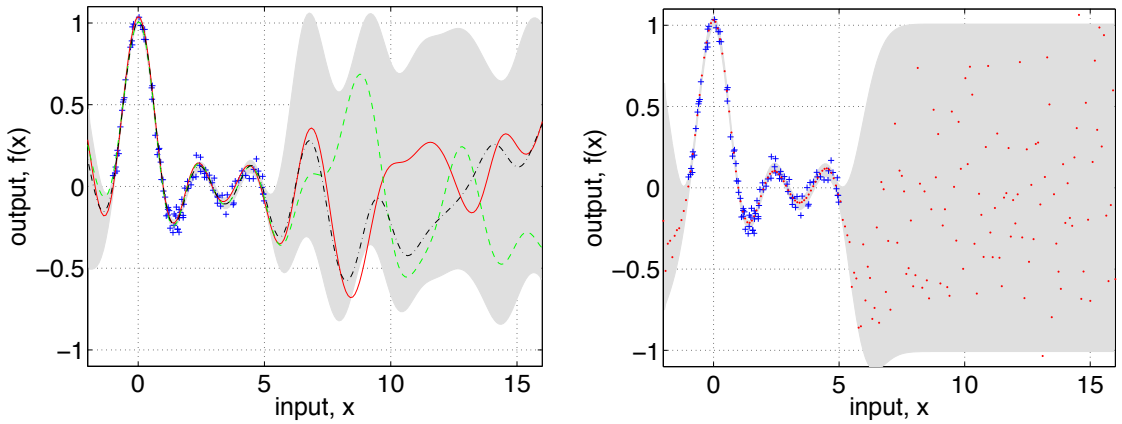
In this section we investigate properties of the SSGP algorithm, and evaluate the computational complexity vs. accuracy tradeoff. We first relate the FITC and SSGP approximations. We then present empirical comparisons on several data sets, using FITC and SMGP as benchmarks. Finally, we revisit the alternative more compact representation of SSGP using phases, and discuss a data set where SSGP performs badly.

Our implementation of SSGP in matlab is available from `http://www.tsc.uc3m.es/~miguel/simpletutorialssgp.php` together with a simple usage tutorial and the data sets from this section. An implementation of FITC is available from Snelson's web page at `http://www.gatsby.ucl.ac.uk/~snelson` .

5.1 Comparing Predictive Distributions for SSGP and FITC

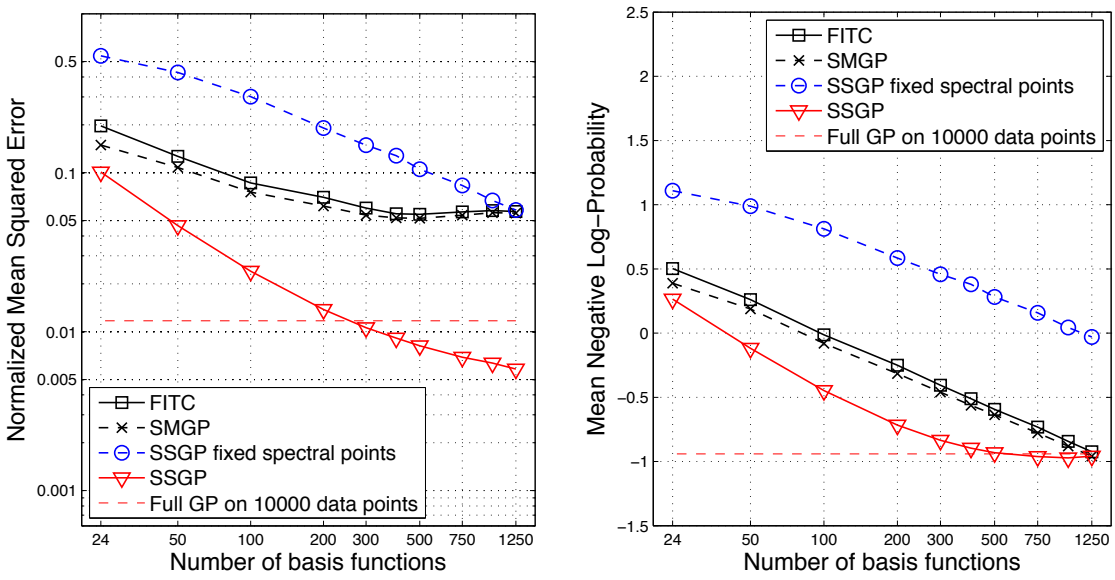
Whereas SSGP relies on a sparse approximation to the spectrum, the FITC approximation is sparse in a spatial sense: A set of *pseudo-inputs* is used as an information bottleneck. The only evaluations of the covariance function allowed are those involving a function value at a pseudo-input. For a set of m pseudo-inputs the computational complexity of FITC is of the same order as that of SSGP with m spectral points.

The covariance function induced by FITC has a constant prior variance, but it is not stationary. The original covariance of the full GP is only approximated faithfully in the vicinity of the pseudo-inputs and the covariance between any two function values that are both far apart from any pseudo-



(a) (b)

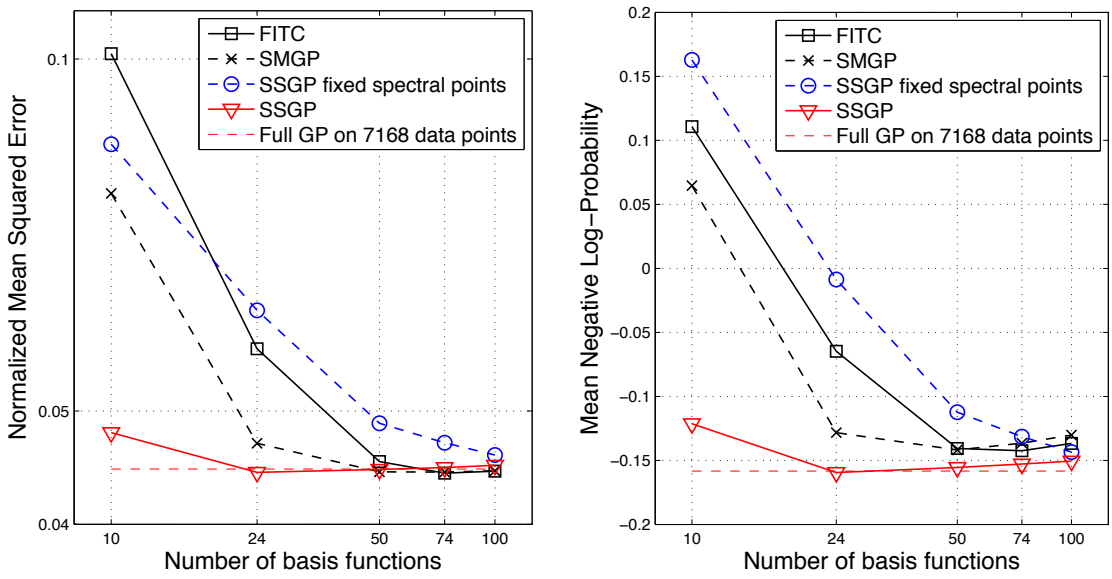
Figure 2: Learning the sinc(x) function from 100 noisy observations (plusses) using 40 basis functions with shaded area showing 95% (noise free) posterior confidence area. In panel (a) the SSGP method with three functions drawn from the posterior is shown. In (b) the same data for the FITC method with samples (dots) drawn from the joint posterior.



(a) (b)

Figure 3: *Kin-40k* data set. (a) NMSE and (b) MNLP as a function of the number of basis functions.

input decays to zero. As a result, functions sampled from the GP prior induced by FITC tend to white Gaussian noise away from the pseudo-inputs.



(a) (b)

Figure 4: *Pumadyn-32nm* data set. (a) NMSE and (b) MNLP as a function of the number of basis functions.

Figure 2 compares the predictive posterior distributions of SSGP and FITC for a simple synthetic data set. The training data is generated by evaluating the sinc function on 100 random inputs $x \in [-1,5]$ and adding white, zero-mean Gaussian noise of variance $\sigma_n^2 = 0.05^2$. SSGP is given 20 fixed spectral points sampled from the spectrum of a squared exponential covariance function, and FITC is given 40 fixed pseudo-inputs sampled uniformly from the range of the training inputs. The rest of the hyperparameters are optimized in both cases by maximizing the marginal likelihood. We plot the 95% confidence interval for both predictive distributions (mean $\pm$ two standard deviations), and draw three samples from the SSGP posterior and one sample from the FITC posterior.

Despite the different nature of the approximations, the figure shows that for an equal number of basis functions both predictive distributions are qualitatively very similar: the uncertainty grows away from the training data. In the following section, we verify empirically that the SSGP is a practical approximation for modelling non-periodic data.

5.2 Performance Evaluation

We will use two quantitative performance measures: the test Normalized Mean Square Error (NMSE) and the test Mean Negative Log Probability (MNLP) , defined as:

$$\text{NMSE} = \frac{\langle (y_{*j} - \mu_{*j})^2 \rangle}{\langle (y_{*j} - \bar{y})^2 \rangle} \quad \text{and} \quad \text{MNLP} = \frac{1}{2} \left\langle \left(\frac{y_{*j} - \mu_{*j}}{\sigma_{*j}} \right)^2 + \log \sigma_{*j}^2 + \log 2\pi \right\rangle , \tag{13}$$

where μ_{*j} and σ_{*j}^2 are, respectively, the predictive mean and variance for test sample j and y_{*j} is the actual test value for that sample. The average output value for training data is $\bar{y}$. We denote

the average over test cases by $\langle \cdot \rangle$. For all experiments the values reported are averages over ten repetitions.

For each data set we report the performance of five different methods: first, the SSGP algorithm as presented in Section 4.2; second, a version of SSGP where the spectral points are "fixed" to samples from the spectral density of a squared exponential covariance function whose lengthscales are learned (SSGP fixed spectral points);[4] third, the FITC approximation, learning the pseudo-inputs; fourth, SMGP, trained as described in Walder et al. (2008); and finally as a base line comparison we report the result of a full GP trained on the entire training set. We plot the performance as a function of the number of basis functions. For FITC this is equivalent to the number of pseudo-inputs, whereas for SSGP a spectral point corresponds to two basis functions. The number of basis functions is a good proxy for computational cost.

We consider four data sets of size moderate enough to be tractable by a full GP, but still large enough that there is a motivation for computationally efficient approximations.

The two first data sets are both artificially generated using a robot arm simulator and are highly non-linear and have very low noise. They were both used in Seeger et al. (2003) and Snelson and Ghahramani (2006), but note that their definition of the NMSE measure differs by a factor of 2 from our definition in (13). We follow precisely their preprocessing and use the original splits. The first data set is *Kin-40k* (8 dimensions, 10000 training and 30000 testing samples) and the results are displayed in Figure 3. For both error measures SSGP outperforms FITC and SMGP by a large margin, and even improves on the performance of the full GP. The SSGP with fixed spectral points is inferior, proving that a greater sparsity vs. accuracy tradeoff can be achieved by optimizing the spectral points.

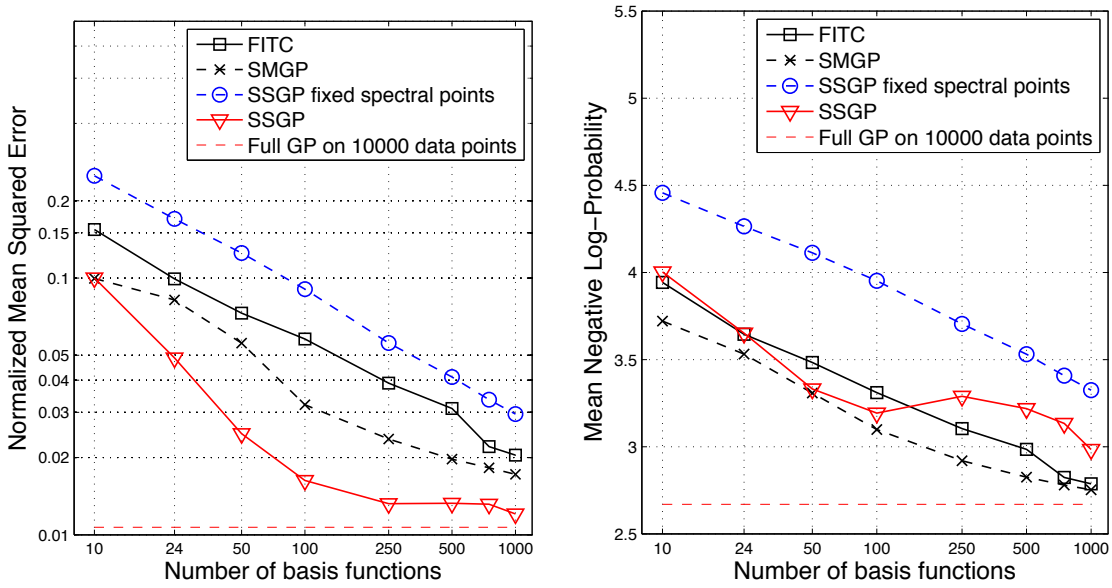
The *Pumadyn-32nm* problem (32 dimensions, 7168 training and 1024 testing samples) can be seen as a test of the ARD capabilities of a regression model, since only 4 out of the 32 input dimensions are relevant. Following Snelson and Ghahramani (2006), to avoid getting stuck at an undesirable bad local optimum, lengthscales are *initialized* from a full GP on a subset of 1024 training data points, for all compared methods. The results are shown in figure 4.

The conclusions are similar as for the *Kin-40k* data set. SSGP matches the full GP for a surprisingly small number of basis functions.

The *Pole Telecomm* and the *Elevators* data sets are taken from `http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html` . In the *Pole Telecomm* data set we retain 26 dimensions, removing constants. We use the original split, 10000 data for training and 5000 for testing. Both the inputs and the outputs take a discrete set of values. In particular, the outputs take values between 0 and 100, in multiples of 10. We take into account the output quantization by lower bounding the value of σ_n^2 to the value of the quantization noise, $\text{bin}_{\text{spacing}}^2/12$. This lower bounding is applied to all the compared methods. The effect is to provide a better estimation for σ_n^2 and therefore, better MNLP measures, but we have observed that this modification has no noticeable effect on NMSE values. Resulting plots are in Figure 5.

SSGP is superior in terms of NMSE, getting very close to the full GP for more than 200 basis functions. In terms of MNLP, SSGP is between FITC and SMGP for small m, but slightly worse for more than 100 basis functions. This may be an indication that SSGP produces better predictive means than variances. We also see that SSGP with fixed spectral points is uniformly worse.

4. In practice the spectral points are sampled from the spectral density of a squared exponential covariance function, and scaled as the lengthscales adapt.



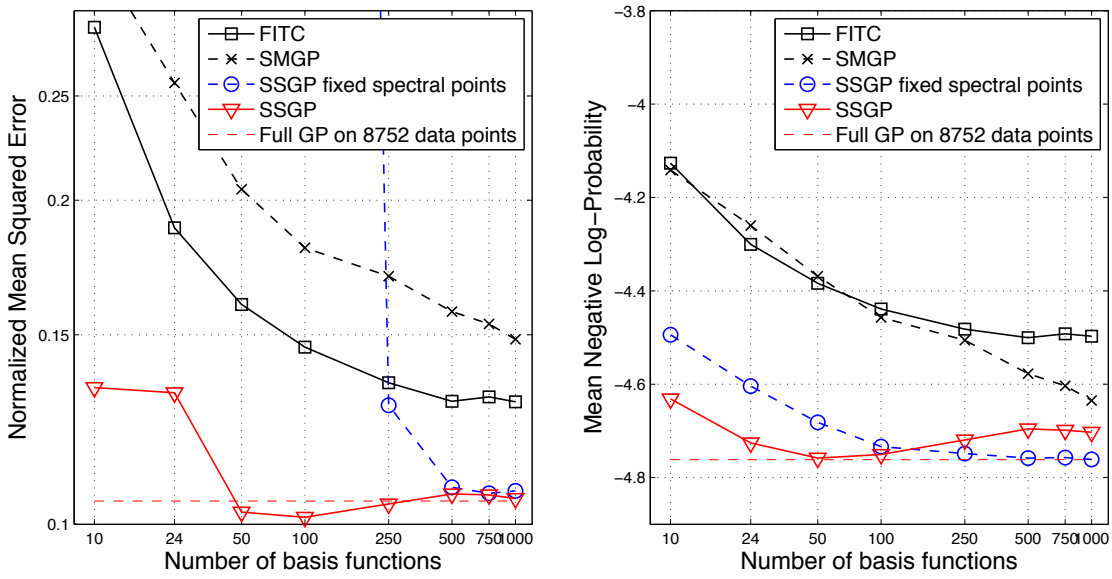
(a) (b)

Figure 5: *Pole Telecomm* data set. (a) NMSE and (b) MNLP as a function of the number of basis functions.

The fourth data set, *Elevators*, relates to controlling the elevators of an F16 aircraft. After removing some constant inputs the data is 17-dimensional. We use the original split with 8752 data for training and 7847 for testing. Results are displayed in Figure 6. SSGP consistently outperforms FITC and SMGP and gets very close to the full GP using a very low number of basis functions. The large NMSE average errors incurred by SSGP with fixed spectral points for small numbers of basis functions are due to outliers that are present in a small number (about 10 out of 7847) of the test inputs, in some of the 10 repeated runs. The predictive variances for these few points are also big, so their impact on the MNLP score is small. Such an effect has not been observed in any of the other data sets.

5.3 Explicit Phase Representation

In Section 3.2 we considered an alternative representation of the SSGP model using only half the basis functions, but explicitly representing the phases. Bayesian inference in this representation is intractable, but one can optimize the phases instead, at the possibly increased risk of overfitting. As an example, we evaluate the performance of the cosine only expansion with explicit phases on the *Pole-Telecomm* data set in Figure 7. Whereas the performance for the two variants are comparable for small numbers of basis functions, the cosine only representation becomes worse when the number of basis functions gets larger, confirming our suspicion that optimization of the phases increases the risk of overfitting.



(a) (b)

Figure 6: *Elevators* data set. (a) NMSE and (b) MNLP as a function of the number of basis functions.

5.4 The Pendulum Data Set

So far we have seen data sets where SSGP consistently outperforms FITC and SMGP, and often approaches the performance of a full GP for quite small numbers of basis functions. In this section we present a counter example, showing that SSGP may occasionally fail, although we suspect that this is the exception rather than the norm.

The small data set *Pendulum* (9 dimensions, 315 training and 315 testing samples) represents the problem of predicting the change in angular velocity of a simulated mechanical pendulum over a short time frame (50 ms) as a function of various parameters of the dynamical system. The target variable depends heavily on all inputs and the targets are almost noise free. Figure 8 shows the results of our experiments. Note that we use up to 800 basis functions for investigation, although for computational reasons it would make sense to use the full GP rather than an approximation with more than 315 basis functions. Although the SSGP NMSE performance is good, we see that especially for large number of basis functions, the MNLP performance is spectacularly bad. A closer inspection shows that the mean predictions are quite accurate, the predictive variances are excessively small. This SSGP model thus exhibits overfitting in the form of being overconfident. Note, that the SSGP with fixed spectral points seems to suffer much less from this effect, as would be expected. Interestingly, re-running the SSGP algorithm with different random initializations gives very different predictions, the predictive distributions from separate runs disagreeing wildly. One could perhaps diagnose the occurrence of the problem in this way. The bottom line is that any algorithm which optimizes the marginal likelihood over a large number of parameters, will risk

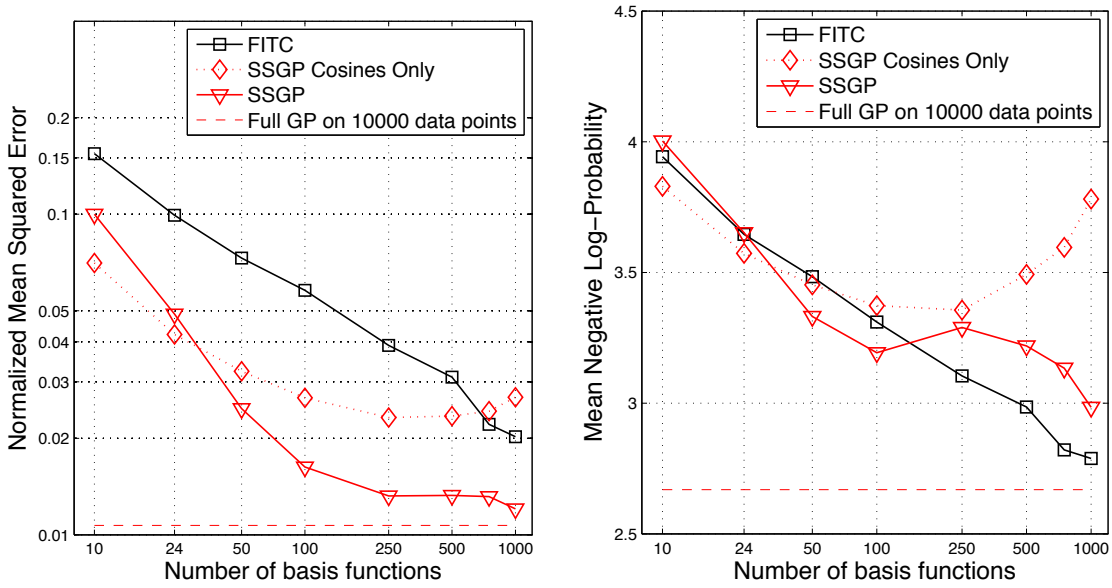


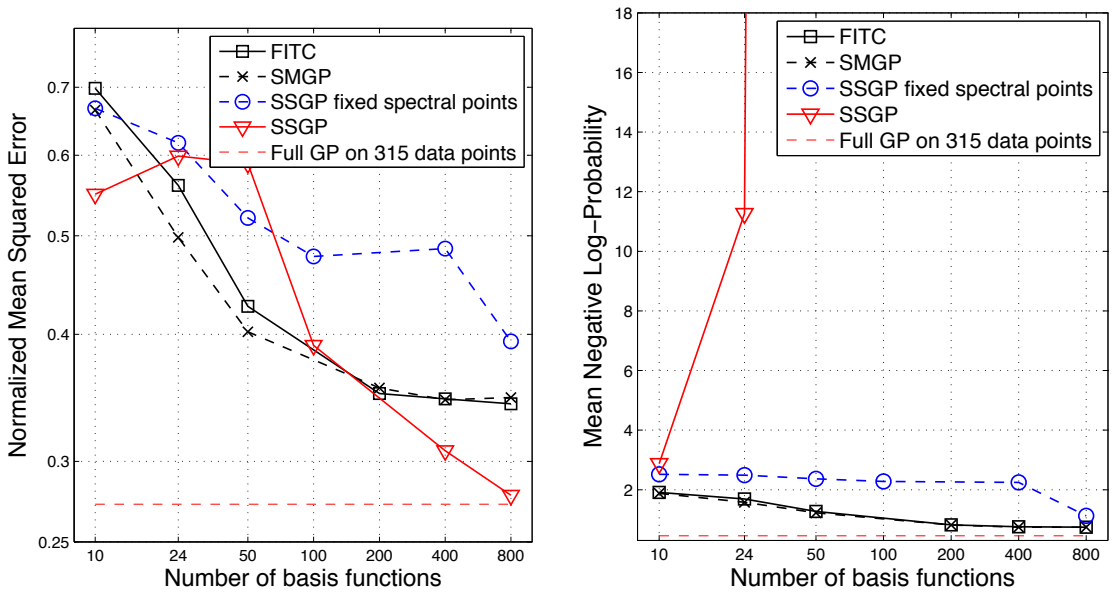
Figure 7: *Pole Telecomm* data set. (a) NMSE and (b) MNLP as a function of the number of basis functions, comparing SSGP with the version with cosines only and explicit phases.

falling in the overfitting trap. We nevertheless think that the SSGP algorithm will often have very good performance, and will be a practically important algorithm, although one must use it with care.

6. Discussion

We have introduced the Sparse Spectrum Gaussian Process (SSGP) algorithm, a novel perspective on sparse GP approximations where rather than the usual sparsity approximation in the spatial domain, it is the spectrum of the covariance function that is subject to a sparse approximation by means of a discrete set of samples, the spectral points. We have provided a detailed comparison of the computational complexity vs. accuracy tradeoff of SSGP to that of the state of the art GP sparse approximation FITC and its extension SMGP. SSGP shows a dramatic improvement in four commonly used benchmark regression data sets, including the two data sets used for evaluation in the paper where FITC was originally proposed (Snelson and Ghahramani, 2006). However, we found a small data set where SSGP badly fails, with good predictive means but with overconfident predictive variances. This indicates that although SSGP is practically a very appealing algorithm, care must be taken to avoid the occasional risk of overfitting.

Other algorithms, such as the variational approach of Titsias (2009) which focus on approaching the full GP in the limit of large numbers of basis functions are to a large degree safeguarded from overfitting. However, algorithms derived from GPs whose focus is on achieving good predictive accuracy on a limited computational budget, such as FITC, SMGP and the currently proposed SSGP,



(a) (b)

Figure 8: *Pendulum* data set. (a) NMSE and (b) MNLP as a function of the number of basis functions.

typically achieve superior performance, see Figure 3 in Titsias (2009), with some risk of overfitting. Note, that these algorithms don't generally converge toward the full GP.

An equivalent view of SSGP is as a sparse Bayesian linear combination of pairs of trigonometric basis functions, a sine and a cosine for each spectral point. The weights are integrated out, and at the price of having two basis functions per frequency, the phases are effectively integrated out as well. We have shown that although a representation in terms of a single basis function per frequency and an explicit phase is possible, learning the phases poses an increased risk of overfitting. If the spectral points are sampled from the power spectrum of a stationary GP, then SSGP approximates its covariance function. However, much sparser solutions can be achieved by learning the spectral points, which effectively implies learning the covariance function. The SSGP model is to the best of our knowledge the only sparse GP approximation that induces a stationary covariance function.

SSGP has been presented here as a Gaussian process prior for regression with a tractable likelihood function from the assumption of Gaussian observation noise. Extending to other types of analytically intractable likelihood functions, such as sigmoid for classification or Laplace for robust regression is possible by using the same approximation techniques as for full GPs. An example is the use of Expectation Propagation in the derivation of generalized FITC (Naish-Guzman and Holden, 2008). Further modifications and extensions of SSGP are discussed in Lázaro-Gredilla (2010).

The main differences between SSGP and most previous approaches to sparse GP regression is the stationarity of the prior and the non-local nature of the basis functions. It will be interesting to

investigate more carefully in the future the exact conditions under which these spectacular sparsity vs. accuracy improvements can be expected.

Acknowledgments

This work has been partly supported by an FPU grant (first author) from the Spanish Ministry of Education and CICYT project TEC-2005-00992 (first and last author). Part of this work was developed while the first author was a visitor at the Computational and Biological Learning Lab, Department of Engineering, University of Cambridge.

Appendix A. Details of the Implementation

In practice, to improve numerical accuracy and speed, Equations (7) and (8) should be implemented using the Cholesky decomposition $\mathbf{R} = \text{chol}(\mathbf{A})$. Thus the predictive distribution is computed as

$$\mathbb{E}[y_*] \;=\; \phi(\mathbf{x}_*)^\top \mathbf{R}\backslash(\mathbf{R}^\top\backslash(\Phi_\mathbf{f}\mathbf{y})) \qquad \mathbb{V}[y_*] \;=\; \sigma_n^2 + \sigma_n^2||\mathbf{R}^\top\backslash\phi(\mathbf{x}_*)||^2,$$

and the log evidence as

$$\log p(\mathbf{y}|\theta) \;=\; -\frac{1}{2\sigma_n^2}\left[||\mathbf{y}||^2 - ||\mathbf{R}^\top\backslash(\Phi_\mathbf{f}\mathbf{y})||^2\right] - \frac{1}{2}\sum_i \log \mathbf{R}_{ii}^2 + m\log\frac{m\sigma_n^2}{\sigma_0^2} - \frac{n}{2}\log 2\pi\sigma_n^2\,,$$

where $\mathbf{R}_{ii}$ refers to the diagonal elements of $\mathbf{R}$.

References

G. L. Bretthorst. Nonuniform sampling: Bandwidth and aliasing. In *Maximum Entropy and Bayesian Methods*, pages 1–28. Kluwer, 2000.

A. B. Carlson. *Communication Systems*. McGraw-Hill, 3rd edition, 1986.

L. Csató and M. Opper. Sparse online Gaussian processes. *Neural Computation*, 14(3):641–669, 2002.

M. Lázaro-Gredilla. *Sparse Gaussian Processes for Large-Scale Machine Learning*. PhD thesis, Universidad Carlos III de Madrid, 2010. URL `http://www.tsc.uc3m.es/~miguel/publications.php`.

M. Lázaro-Gredilla and A.R. Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems 22*, pages 1087–1095. MIT Press, 2010.

M. Lázaro-Gredilla, J. Quiñonero-Candela, and A. Figueiras-Vidal. Sparse spectral sampling Gaussian processes. Technical report, Microsoft Research, 2007.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

A. Naish-Guzman and S. Holden. The generalized FITC approximation. In *Advances in Neural Information Processing Systems 20*, pages 1057–1064. MIT Press, 2008.

J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

J. Quiñonero-Candela, C. E. Rasmussen, and C. K. I. Williams. Approximation methods for Gaussian process regression. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*, pages 203–223. MIT Press, 2007.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pages 1177–1184. MIT Press, Cambridge, MA, 2008.

C. E. Rasmussen and Joaquin Quiñonero-Candela. Healing the relevance vector machine through augmentation. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 689–696, 2005.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *Proceedings of the 9th International Workshop on AI Stats*, 2003.

A. J. Smola and P. Bartlett. Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1259Ű–1266. MIT Press, 2006.

M. L. Stein. *Interpolation of Spatial Data*. Springer Verlag, 1999.

M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the 12th International Workshop on AI Stats*, 2009.

V. Tresp. A Bayesian committee machine. *Neural Computation*, 12:2719–2741, 2000.

R. Urtasun and T. Darrell. Sparse probabilistic regression for activity-independent human pose inference. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.

C. Walder, K. I. Kim, and B. Schölkopf. Sparse multiscale Gaussian process regression. In *25th International Conference on Machine Learning*. ACM Press, New York, 2008.

C. K. I. Williams. Computing with infinite networks. In *Advances in Neural Information Processing Systems 9*, pages 1069–1072. MIT Press, 1997.

C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

Journal of Machine Learning Research 11 (2010) 1883-1926 Submitted 1/10; Published 6/10

Fast and Scalable Local Kernel Machines

Nicola Segata SEGATA@DISI.UNITN.IT
Enrico Blanzieri BLANZIER@DISI.UNITN.IT
Department of Information Engineering and Computer Science
University of Trento
Trento, Italy

Editor: Léon Bottou

Abstract

A computationally efficient approach to local learning with kernel methods is presented. The **Fa**st **L**ocal **K**ernel **S**upport **V**ector **M**achine (FaLK-SVM) trains a set of local SVMs on redundant neighbourhoods in the training set and an appropriate model for each query point is selected at testing time according to a proximity strategy. Supported by a recent result by Zakai and Ritov (2009) relating consistency and localizability, our approach achieves high classification accuracies by dividing the separation function in local optimisation problems that can be handled very efficiently from the computational viewpoint. The introduction of a fast local model selection further speeds-up the learning process. Learning and complexity bounds are derived for FaLK-SVM, and the empirical evaluation of the approach (with data sets up to 3 million points) showed that it is much faster and more accurate and scalable than state-of-the-art accurate and approximated SVM solvers at least for non high-dimensional data sets. More generally, we show that locality can be an important factor to sensibly speed-up learning approaches and kernel methods, differently from other recent techniques that tend to dismiss local information in order to improve scalability.

Keywords: locality, kernel methods, local learning algorithms, support vector machines, instance-based learning

1. Introduction

Efficiently processing large amount of data is one of the challenges of current research in kernel methods. Although most of the recently proposed techniques are based on different approaches, their common assumption is that scalability can be obtained by limiting or reducing the complexity of the decision function. In fact, very fast training algorithms have been developed for linear SVM (Keerthi and DeCoste, 2005; Collins et al., 2008; Chang et al., 2008; Bordes et al., 2009; Fan et al., 2008), and indeed they are effective when the linear separation is a good choice such as in high-dimensionality problems. Other approaches permit the non-linear feature space setting, but they limit the complexity by working with a reduced number of examples or a small set of support vectors (Lee and Mangasarian, 2001), using active and online example selection (Bordes et al., 2005; Bordes and Bottou, 2005) or bounding the number of basis functions (Keerthi et al., 2006; Joachims and Yu, 2009).

In the works referenced above, computational efficiency is sought bounding some aspects of the optimisation problem. The result is an *approximation* of the optimal separation and a *smoothing* of the decision function which is more influenced by the global distribution of the examples than by the local behaviour of the unknown target function in each specific sub-region. The emerging approach

is thus to trade locality for scalability permitting, with a potentially high level of under-fitting, to achieve a fast convergence to an approximated solution of the optimisation problem.

We show here that locality is not necessary related to computational inefficiency, but, instead, it can be the key factor to obtain very fast kernel methods without the need to smooth locally the global decision function. In our proposed approach, the model is formed by a set of accurate local models trained on fixed-cardinality sub-regions of the training set and the prediction module uses for each query point the more appropriate local model. In this setting, we are not approximating with some level of inaccuracy the original SVM optimisation problem, but we are separately considering different parts of the decision function with the potential advantage of better capturing the local separation. So, instead of locally under-fit the decision function by globally smoothing it like approximated SVM solvers do, we search for decision functions that are locally-calculated and they are very similar (or even better) in terms of accuracy to the global decision function in the proximity of each testing point. This approach is theoretically supported also by the recent result obtained by Zakai and Ritov (2009) that showed how, roughly speaking, "consistency implies local behaviour".

In this work we present **Fa**st **L**ocal **K**ernel **S**upport **V**ector **M**achine (FaLK-SVM), that pre-computes a set of local SVMs covering with adjustable redundancy the whole training set and uses for prediction a model which is the nearest (in terms of neighbourhood rank in feature space) to each testing point. FaLK-SVM is obtained introducing various strategies, detailed below, to speed-up the Local SVM approach (see Blanzieri and Melgani, 2006 and Section 3.3). Scalability is obtained approximating the Local SVM approach softening the assumption that the query point must be the central example of the neighbourhood on which the local SVM is trained; in this way we use the same local SVM model for more than one testing point and we can also pre-compute the local models during training. The locality of the approach is regulated by the neighbourhood size k and the method uses all the training points. Starting from the theory of local learning algorithms (Bottou and Vapnik, 1992; Vapnik and Bottou, 1993) we derive generalisation bounds for FaLK-SVM, and we analyse the computational complexity stating that, under reasonable assumptions, the training of our technique scales as $N \log N$ and the testing as $\log N$ where N is the training set size. We also introduce a procedure for local model selection in order to speed-up the selection of the parameters and better capturing local properties of the data. The empirical evaluation (with data sets with up to 3 million examples) shows that FaLK-SVM outperforms accurate and approximated SVM solvers both in term of generalisation accuracy and computational performances.

The effectiveness and efficiency of our approach is directly related to the role that locality plays in the learning problem. It is well known, for example, that for very high-dimensional problems such as text and document classification, the linear kernel performs better than non-linear kernels which are hard to tune and can be subject to the "curse of dimensionality" (Bengio et al., 2005). On the other hand, there are problems (Blackard and Dean, 1999; Uzilov et al., 2006) which inherently require non-linear approaches to be tackled. This is due to the combination of an intrinsic dimensionality which is low with respect to the training set size and of a decision function which is not simple to learn. In general, locality plays a more important role as the number of training examples increases because the ratio between training set cardinality and the dimensionality is more favourable and the local characteristics are more evident. Other signals for the need of a non-linear kernel are the detection of uneven distributions in the data sets (typical of real-world problems), the monotonic increasing of accuracy with respect to training size also for already large amount of data and the inclusion of a high fraction of training examples in the support vector set. A representative

of this class of problems is the Forest CoverType data set (Blackard and Dean, 1999) which is a large real data set (more than half a million examples) with bounded dimensionality (54 features) that needs as many examples as possible to increase accuracy. We already showed in a very preliminary study (Segata and Blanzieri, 2009c) that our approach on this data set is more accurate than SVM and much faster than both accurate and approximated SVM solvers.

The present contribution can be seen from multiple viewpoints. (i) FaLK-SVM modifies the Local SVM approach (Blanzieri and Melgani, 2006; Zhang et al., 2006) that showed excellent classification performances but had dramatic computational problems, leading to a scalable Local SVM classifier asymptotically much faster than SVM. (ii) The approach is also an enhancement of the local learning algorithms because the learning process is not delayed until the prediction phase (*lazy learning*) but the construction of the local models occurs during training (*eager learning*). (iii) From a practical viewpoint, FaLK-SVM is a novel kernel method which outperforms accurate and approximated SVM solvers for non high-dimensional data sets. (iv) For complex classification problems that require an high fraction of support vectors (SVs), we exploit locality to avoid the need of bounding the number of total SVs as existing approximated SVM solvers do for computational reasons. (v) More generally, our approach can also be seen as a framework for localising and make scalable any kernel method, classifier and regressor and in general every data analysis that can be applied on sub-regions of the entire data set. The proposed FaLK-SVM classifier and related tools are freely available with source code as part of the Fast Local Kernel Machine Library (Segata, 2009, FaLKM-lib).

In the next Section we analyse the work on local learning algorithms, Local SVM and fast large margin classifiers that are all related with our work. Section 3 formally introduces some machine learning tools that we need in order to introduce FaLK-SVM in Section 4 and analyse its learning bounds, complexity bounds, implementation, local model selection procedure and intuitive interpretation. Section 5 details the empirical evaluation with respect to accurate and approximated approaches.

2. Related Work

Locality is often a crucial component of machine learning systems, although we are not aware of approaches exploiting locality for improving the computational performances. We review in this section those areas that are more related with our approach: local learning algorithms, local support vector machines, approximated and scalable SVM solvers.

2.1 Local Learning Algorithms

Local learning algorithms (LLAs) are a class of learning approaches introduced by Bottou and Vapnik (1992). Instead of estimating a decision function which is optimal (with respect to some criteria) for all possible unseen testing examples, the idea underlying LLAs consists in estimating the optimal decision function for each single testing point. The value of the function is estimated in a small sub-region of the input space around the query point. For a local learning algorithm, the points in the proximity of the query point have an higher influence in the training of the local model. The approach is particularly effective for uneven distributed data sets, that is, data sets presenting regions in which the examples have different spatial resolutions. In fact, with LLAs, the characteristics of the learning process can be locally adjusted. A proper choice of the locality parameter can reduce the generalisation error with respect to a global classifier as formalised by the

Local Risk Minimization principle (Vapnik and Bottou, 1993; Vapnik, 2000). Notice that there are various ways of specifying the degree of locality for LLAs as discussed for instance by Atkeson et al. (1997). Examples of LLAs are the well-known k-Nearest Neighbours (kNN) classifier, the Radial Basis Function networks (Broomhead and Lowe, 1988), and the Local SVM classifier (Blanzieri and Melgani, 2006; Zhang et al., 2006) described in Section 2.2.

Despite their theoretical and practical appeal, LLAs seem not to have been studied in depth in the last few years. This is probably due to the fact that LLAs, as formulated by Bottou and Vapnik (1992), fall in the class of *lazy learning* (or *memory-based learning*) that have great overhead on the testing phase, as opposed to *eager learning* in which the function estimation is performed during training increasing the computational performances of the testing phase.

2.2 Local Support Vector Machines

The main idea of Local SVM, described in details in Section 3.3, is to build at prediction time an example-specific maximal marginal hyperplane based on the set of k-neighbours.

Local SVM is a LLA and was independently proposed by Blanzieri and Melgani (2006, 2008) and by Zhang et al. (2006) and applied respectively to remote sensing and visual recognition tasks. Other successful applications of the approach are detailed by Segata and Blanzieri (2009a) for general real data sets, by Blanzieri and Bryl (2007) for spam filtering and by Segata, Blanzieri, Delany, and Cunningham (2009b) for noise reduction. Similar approaches have been presented by Yang and Kecman (2008) and applied in the medical domain (Yang and Kecman, 2009) and for face recognition problems (Yang and Kecman, 2010).

However, Local SVM suffers from the high computational cost of the testing phase that comprises, for each example, (i) the selection of the k nearest neighbours and (ii) the computation of the maximal separating hyperplane on the k examples. An attempt to computationally improve the Local SVM approach of Zhang et al. (2006) has been proposed by Cheng et al. (2007) where the idea is to train multiple SVMs on clusters found by a variant of k-means, called MagKmeans, that introduces in the clustering criterion the requirement that the clusters cannot have unbalanced class cardinalities. However the method does not follow directly the idea of Local SVM, the main difference being that it can build only local linear models and the size of the clusters is not fixed (MagKmeans does not have constraints on the cardinalities and the balancing requirement can cause the detection of clusters with high cardinalities). The achieved computational performances are better than their formulation of Local SVM, but worse than global SVM.

2.3 Fast Large Margin Classifiers

The need for fast and scalable kernel-based classifiers led to the development of several methods in the last few years, although considerable attention seems to have been focused especially on linear SVM classifiers. Below, we initially consider the works applicable also to non-linear kernels, successively we review the works on the linear case.

One of the first large-scale maximal margin learning that can use non-linear kernel functions is represented by Core Vector Machines (Tsang et al., 2005, CVM); reformulating the SVM approach as a minimum enclosing ball problem, the authors proved that it is possible to obtain approximated optimal solution in competitive training times by using the core sets. Good results have been achieved using non-linear kernels although it has been pointed out that the choice of the stopping criteria is crucial for the trade-off between computational efficiency and generalisation accuracy.

Ball Vector Machines (Tsang et al., 2007, BVM) are a modification of CVM in which the minimality of the enclosing balls is not required, because the radius of the ball is fixed. The resulting classifier improves the computational performances. Another approach based on an online setting of the SVM optimisation problem has been proposed by Bordes et al. (2005, LASVM) and by Bordes and Bottou (2005) and it is an algorithm that converges to the SVM solution. It has been shown that competitive accuracies can be achieved also after a single pass over the training set. The approach can be seen as a SVM solver that includes a support vector removal step. In addition, several strategies for active training-points selection can further improve computational and generalisation performances. Formulating the optimisation problem in the primal, Keerthi et al. (2006, SpSVM) proposed a method that bounds the number of basis functions considered and thus the computational complexity. Increasing the cardinality of the basis function set allows the method to converge to the SVM solution. A greedy strategy guides the choice of the basis functions to be included in the working set. Collobert et al. (2006, USVM) showed that softening the convex setting of maximal margin classifiers using a non-convex loss function can bring computational advantage over the corresponding standard convex problem. The non-convex problem is solved using the *concave-convex procedure* (Yuille and Rangarajan, 2003). Recently, the Cutting-Plane Subspace Pursuit (Joachims and Yu, 2009, CPSP) based on cutting-plane training (Joachims et al., 2009) has been proposed; it permits to learn maximal-margin decision functions in the feature space using arbitrary basis vectors instead of the support vectors only. This can results in sparser solutions increasing the testing and training computational performances especially for high-dimensional data sets. Although not always considered a method for large-scale learning, LibSVM (Chang and Lin, 2001) demonstrated to be competitive with approximated approaches from the computational viewpoint. LibSVM is a SVM solver implementing a SMO-type decomposition method proposed by Fan et al. (2005) integrating it with caching and shrinking (Joachims, 1999).

Large margin classifiers can also achieve scalability using subsampling-based approaches that train the model on a relatively small subset of the whole training set. However, the accuracy of SVM with subsampling can decrease due to the loss of information contained in the discarded training points. The decreasing of accuracy with respect to SVM without subsampling is more dramatic when a complex decision function is needed. In these cases the accuracy problems can be mitigated or reduced by developing an ensemble of classifiers. Bootstrap aggregating (bagging) by Breiman (1996) is an effective strategy to perform accurate classification using an ensemble of classifiers trained on subsets of the training set (using uniform sampling with replacement) that can also overcome the accuracies of SVM. Bagging with SVM can thus be used for obtaining scalability as long as the advantage of training smaller SVM models on subsets of the training set (that can scale cubically) overcome the disadvantage of training multiple SVMs.

Recently a lot of work has been performed in order to develop very fast and scalable solvers applicable to *linear* SVM only. Keerthi and DeCoste (2005) modified the Finite Newton method of Mangasarian (2002) introducing robust conjugate gradient techniques and other heuristics. Joachims (2006) developed an alternative formulation of the SVM optimisation problem exploiting a different form of sparsity. Lin et al. (2007) used logistic regression with Trust Region Newton Methods. Variants of coordinate descent methods for linear SVM are developed by Chang et al. (2008) in the primal and by Hsieh et al. (2008) in the dual. A different gradient approach was developed by Smola et al. (2008). Other approaches are based on Stochastic Gradient Descent (SGD) like those developed by Shalev-Shwartz et al. (2007) and by Bordes et al. (2009) which work in the primal, whereas Collins et al. (2008) apply SGD in the dual. Although SGD methods can be theoretically

used for non-linear SVM the performances are analysed for the linear case only. LIBLINEAR (Fan et al., 2008) is a fast software package implementing some of the cited works. The common idea of all the proposed methods is that the advantage of having a method that uses a huge number of training points overcomes the disadvantage of approximating the decision function with a linear model. This is effective, as explicitly noticed in almost all the cited works, when the dimensionality is very large and thus the problem is very sparse. This is, for example, the typical situation of text document classification. However, when the needed decision function is highly non-linear and the intrinsic dimensionality of the space is relatively small, the linear SVM approach cannot compete with SVM using non-linear kernels in terms of generalisation accuracy. Apart from the generalisation ability also the computational performances can be compromised in these cases, because the algorithm cannot find a good decision function and so convergence problems can occur.

3. Preliminaries

In order to introduce our approach, we need to analyse the formulation of kNN, SVM, *k*NNSVM and cover trees.

Here and in the following of the paper, we consider a binary class classification with examples $(\mathbf{x}_i, y_i) \in \mathcal{H} \times \{-1,+1\}$ for $i = 1,\ldots,N$ and $\mathcal{X} = \{\mathbf{x}_i \,|\, i = 1,\ldots,N\}$, where $\mathcal{H}$ is an Hilbert space with inner product $\langle \cdot,\cdot \rangle$ and norm $\|\cdot\|$. Extensions to multi-class problems will be explicitly discussed.

3.1 The *k* Nearest Neighbour Algorithm

Given an example $\mathbf{x}' \in \mathcal{H}$, it is possible to order an entire set of points $\mathcal{X}$ with respect to $\mathbf{x}'$. This corresponds to define a function $r_{\mathbf{x}'} : \{1,\ldots,N\} \to \{1,\ldots,N\}$ that recursively reorders the indexes of the N points in $\mathcal{X}$:

$$\begin{cases} r_{\mathbf{x}'}(1) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\mathbf{x}_i - \mathbf{x}'\| \\ r_{\mathbf{x}'}(j) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\mathbf{x}_i - \mathbf{x}'\| \qquad i \neq r_{\mathbf{x}'}(1),\ldots,r_{\mathbf{x}'}(j-1) \quad \text{for} \quad j = 2,\ldots,N. \end{cases}$$

In this way, $\mathbf{x}_{r_{\mathbf{x}'}(j)}$ is the example in the j-th position in terms of distance from $\mathbf{x}'$, namely the j-th nearest neighbour, $\|\mathbf{x}_{r_{\mathbf{x}'}(j)} - \mathbf{x}'\|$ is its distance from $\mathbf{x}'$ and $y_{r_{\mathbf{x}'}(j)}$ is its class. In other terms:

$$j < k \Rightarrow \|\mathbf{x}_{r_{\mathbf{x}'}(j)} - \mathbf{x}'\| \leq \|\mathbf{x}_{r_{\mathbf{x}'}(k)} - \mathbf{x}'\|.$$

Given the above definition, the majority decision rule of kNN for binary classification problems is defined by

$$\text{kNN}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{k} y_{r_{\mathbf{x}}(i)}\right).$$

For problems with more than two classes, the decision rule of kNN is the usual majority rule, namely the method selects the class with the highest number of representatives in the k-neighbourhood instead of taking the sign of the summation.

3.2 Support Vector Machines

SVMs (Cortes and Vapnik, 1995) are classifiers with sound foundations in statistical learning theory (Vapnik, 2000). The decision rule is

$$\mathrm{SVM}(\mathbf{x}) = \mathrm{sign}(\langle w, \Phi(\mathbf{x})\rangle_{\mathcal{F}} + b)$$

where $\Phi(\mathbf{x}) : \mathcal{H} \to \mathcal{F}$ is a mapping in a transformed Hilbert feature space, called $\mathcal{F}$, with inner product $\langle\cdot,\cdot\rangle_{\mathcal{F}}$. The parameters $w \in \mathcal{F}$ and $b \in \mathbb{R}$ are such that they minimise an upper bound on the expected risk while minimising the empirical risk. The minimisation of the complexity term is achieved by the minimisation of the quantity $\frac{1}{2}\cdot\|w\|^2$, which is equivalent to the maximisation of the margin between the classes. In the optimisation problem, the violation of the margin is prevented by the following set of constraints:

$$y_i\left(\langle w, \Phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b\right) \geq 1. \tag{1}$$

If a linear separation cannot be found in the input or feature space, the soft-margin variant of SVM permits the violation of the margin and the presence of misclassified training examples. This is possible introducing slack variables ξ_i (the empirical risk):

$$y_i\left(\langle w, \Phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b\right) \geq 1 - \xi_i \qquad \xi_i \geq 0, i = 1, \ldots, N. \tag{2}$$

For soft-margin SVM the optimisation problem with linear penalisation of ξ_i (L1-norm), becomes the minimisation of $\frac{1}{2}\cdot\|w\|^2 + C\sum_i \xi_i$ subject to (2). Reformulating such an optimisation problem with Lagrange multipliers α_i ($i = 1, \ldots, N$), and introducing a positive definite kernel (PD) function[1] $K(\cdot,\cdot)$ that substitutes the scalar product in the feature space $\langle\Phi(\mathbf{x}_i), \Phi(\mathbf{x})\rangle_{\mathcal{F}}$ the decision rule can be expressed as:

$$\mathrm{SVM}(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right).$$

Throughout this work, SVM denotes the soft-margin SVM.

The kernel trick avoids the explicit definition of the feature space $\mathcal{F}$ and of the mapping function Φ (Schölkopf and Smola, 2002). Popular kernels are the linear kernel, the radial basis function kernel, and the homogeneous and inhomogeneous polynomial kernels. Their definitions are:

$$\begin{aligned} K^{lin}(\mathbf{x},\mathbf{x}') &= \langle \mathbf{x}, \mathbf{x}'\rangle & K^{rbf}(\mathbf{x},\mathbf{x}') &= \exp\left(-\tfrac{\|\mathbf{x}-\mathbf{x}'\|^2}{\sigma}\right), \\ K^{hpol}(\mathbf{x},\mathbf{x}') &= \langle \mathbf{x}, \mathbf{x}'\rangle^d & K^{ipol}(\mathbf{x},\mathbf{x}') &= (\langle \mathbf{x}, \mathbf{x}'\rangle + 1)^d. \end{aligned}$$

The maximal separating hyperplane defined by SVM has been shown to have important generalisation properties and nice bounds on the VC dimension (Vapnik, 2000).

Multiple methods has been proposed in order to apply the maximal margin principle of SVM on multiple class problems. The more popular are the one-against-all method (Bottou et al., 1994) which builds a number of binary decision functions equal to the number of classes N_{cl}, the one-against-one method (Knerr et al., 1990; Kressel, 1999) which builds $N_{cl}\cdot(N_{cl}-1)/2$ binary decision functions using voting in the prediction phase, and the Directed Acyclic Graph SVM (Platt et al.,

1. For convention we refer to kernel functions with the capital letter K and to the number of nearest neighbours with the lower-case letter k.

2000, DAGSVM) which is a modification of the one-against-all method. Other general strategies for reducing the multi-class classification setting to a binary classification problem have been analysed and developed by Allwein et al. (2000). The study carried on by Hsu and Lin (2002) shows that, for SVM, the more effective strategies are the one-against-one and DAGSVM approaches.

3.3 Local SVM: The *k*NNSVM Classifier

We already introduced the idea of Local SVM in Section 2.2, here we detail *k*NNSVM which is the formulation of Local SVM proposed by Blanzieri and Melgani (2006, 2008). *k*NNSVM can be seen as a modification of the SVM approach in order to obtain a LLA able to locally adjust the capacity of the training systems.

In order to classify a given example $\mathbf{x}' \in \mathcal{H}$, we need first to retrieve its k-neighbourhood in the transformed feature space $\mathcal{F}$ and, then, to search for an optimal separating hyperplane only over this k-neighbourhood. In practice, this means that an SVM is built over the neighbourhood in $\mathcal{F}$ of each test example $\mathbf{x}'$. Accordingly, the constraints in (1) become:

$$y_{r_{\mathbf{x}'}(i)} \left(w \cdot \Phi(\mathbf{x}_{r_{\mathbf{x}'}(i)}) + b \right) \geq 1 - \xi_{r_{\mathbf{x}'}(i)}, \text{ with } i = 1, \ldots, k$$

where $r_{\mathbf{x}'} : \{1, \ldots, N\} \rightarrow \{1, \ldots, N\}$ is a function that reorders the indexes of the training examples defined as:

$$\begin{cases} r_{\mathbf{x}'}(1) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}')\|^2_{\mathcal{F}} \\ r_{\mathbf{x}'}(j) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}')\|^2_{\mathcal{F}} \qquad i \neq r_{\mathbf{x}'}(1), \ldots, r_{\mathbf{x}'}(j-1) \quad \text{for} \quad j = 2, \ldots, N. \end{cases} \tag{3}$$

In this way, $\mathbf{x}_{r_{\mathbf{x}'}(j)}$ is the example in the j-th position in terms of distance from $\mathbf{x}'$ and thus $j < k \Rightarrow \|\Phi(\mathbf{x}_{r_{\mathbf{x}'}(j)}) - \Phi(\mathbf{x}')\|_{\mathcal{F}} \leq \|\Phi(\mathbf{x}_{r_{\mathbf{x}'}(k)}) - \Phi(\mathbf{x}')\|_{\mathcal{F}}$ because of the monotonicity of the quadratic operator. The computation is expressed in terms of kernels as:

$$\begin{aligned} ||\Phi(\mathbf{x}) - \Phi(\mathbf{x}')||^2_{\mathcal{F}} &= \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}) \rangle_{\mathcal{F}} + \langle \Phi(\mathbf{x}'), \Phi(\mathbf{x}') \rangle_{\mathcal{F}} - 2 \cdot \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{F}} = \\ &= K(\mathbf{x}, \mathbf{x}) + K(\mathbf{x}', \mathbf{x}') - 2 \cdot K(\mathbf{x}, \mathbf{x}'). \end{aligned}$$

If the kernel is the RBF kernel or any polynomial kernels with degree 1, the ordering function is equivalent to the one defined by the Euclidean metric. In general, for some non-linear kernels (other than the RBF kernel) the ordering function can be quite different to that produced using the Euclidean metric.

The decision rule associated with the method for an example $\mathbf{x}$ is:

$$k\text{NNSVM}(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^{k} \alpha_{r_{\mathbf{x}}(i)} y_{r_{\mathbf{x}}(i)} K(\mathbf{x}_{r_{\mathbf{x}}(i)}, \mathbf{x}) + b \right).$$

For $k = N$, the *k*NNSVM method is the usual SVM whereas, for $k = 2$, the method implemented with the linear or Gaussian radial basis function kernel corresponds to the standard 1-NN classifier. Notice that in situations where the neighbourhood contains only one class the local SVM does not find any separation and so considers all the neighbourhood to belong to the predominant class similarly to the behaviour of the majority rule. Considering *k*NNSVM as a local SVM classifier

built in the feature space, the method has been shown to have a potentially favourable bound on the expectation of the probability of test error with respect to SVM (Blanzieri and Melgani, 2008).

The generalisation of *k*NNSVM for multi-class classification can occur locally, that is solving the local multi-class SVM problem, or globally, that is applying the binary *k*NNSVM classifier on multiple global binary problems. In Segata and Blanzieri (2009a) the adopted strategy for multi-class classification with *k*NNSVM is the one-against-one strategy applied on the local problems. The choice of the one-against-one approach gave good results in comparison with the same strategy on SVM, but no specific empirical studies have been performed yet to identify the most appropriate strategy for multi-class classification with Local SVM.

3.4 Cover Trees

A cover tree is a data structure introduced by Beygelzimer et al. (2006) for performing exact nearest-neighbour operations in a fast and efficient way. Cover trees can be applied in general metric spaces without any other assumption on their structure and thus also in Hilbert spaces calculating the distances by means of kernel functions using the kernel trick.

In more detail, a cover tree can be viewed as a sub-graph of a navigating net (Krauthgamer and Lee, 2004) and it is a levelled tree in which each level (indexed by a decreasing integer i) is a cover (i.e., is representative) for the level beneath it. Every node of a cover tree T is associated with a point of a data set S. Denoting with C_i the set of points associated with nodes in T at level i, with $b > 1$ a constant, and with $dist(\cdot,\cdot)$ the distance function defining the metric of the space, the invariants of a cover tree are:

Nesting $C_i \subset C_{i-1}$

Covering tree For every $\mathbf{p} \in C_{i-1}$ there exists a $\mathbf{q} \in C_i$ such that $dist(\mathbf{p},\mathbf{q}) < b^i$ and the node in level i associated with $\mathbf{q}$ is a parent of the node in level $i-1$ associated with $\mathbf{p}$.

Separation For all distinct $\mathbf{p},\mathbf{q} \in C_i$, $dist(\mathbf{p},\mathbf{q}) > b^i$.

Intuitively, the nesting invariant means that once a point appears in a level, it is present for every lower level. The covering tree invariant implies that every node has a parent in a higher level such that the distance between the respective points is less than b^i, while separation invariant assures that the distance between every pair of points associated to the nodes of a level i is higher than b^i. In addition, the root of the tree (called C_∞ and containing only one example) is a randomly chosen example.

Cover trees have state-of-the-art performance for exact nearest neighbour operations for general metrics in low-dimensional spaces both in terms of computational complexity and space requirements. As theoretically proved by Beygelzimer et al. (2006), the space required by the cover tree data-structure is linear in the data set size ($O(n)$), the computational time of single point insertions, deletions and exact nearest neighbour queries is logarithmic ($O(\log n)$) while the cover tree can be built in $O(n \log n)$.

4. FaLK-SVM: A Fast and Scalable Local Kernel Machine

In this section we introduce our novel technique. Initially we detail the way to pre-compute the local models during training (Section 4.1) and the strategies to reduce the number of local models

(Section 4.2). We then describe the prediction mechanism in Section 4.2.2 and our approach for fast local model selection in Section 4.3. Successively, we derive learning bounds for the approach in Section 4.4 before discussing the computational complexity in Section 4.5 and some details about the implementation (Section 4.6).

4.1 Pre-computing the Local Models during Training Phase

For the local approach we are proposing here, we need to generalise the decision rule of *k*NNSVM to the case in which the local model is trained on the *k*-neighbourhood of a point distinct, in the general case, from the query point. A modified decision function for a query point $\mathbf{q} \in \mathcal{H}$ and another (possibly different) point $\mathbf{t} \in \mathcal{H}$ is:

$$k\text{NNSVM}_{\mathbf{t}}(\mathbf{q}) = \text{sign}\left(\sum_{i=1}^{k} \alpha_{r_{\mathbf{t}}(i)} y_{r_{\mathbf{t}}(i)} K(\mathbf{x}_{r_{\mathbf{t}}(i)}, \mathbf{q}) + b\right) \tag{4}$$

where $r_{\mathbf{t}}(i)$ is the *k*NNSVM ordering function (see above Section 3.3) and $\alpha_{r_{\mathbf{t}}(i)}$ and b come from the training of an SVM on the *k*-neighbourhood of $\mathbf{t}$ in the feature space. In the following we will refer to $k\text{NNSVM}_{\mathbf{t}}(\mathbf{q})$ as being centred on $\mathbf{t}$, to $\mathbf{t}$ as the centre of the model, and, if $\mathbf{t} \in \mathcal{X}$, to $V_{\mathbf{t}}$ as the Voronoi cell induced by $\mathbf{t}$ in $\mathcal{X}$, formally:

$$V_{\mathbf{t}} = \{\mathbf{p} \in \mathcal{H} \text{ s.t. } \|\mathbf{p}-\mathbf{t}\| \leq \|\mathbf{p}-\mathbf{x}\|, \forall \mathbf{x} \in \mathcal{X} \text{ with } t \neq \mathbf{x}\}.$$

The original decision function of *k*NNSVM corresponds to the case in which $\mathbf{t} = \mathbf{q}$, and thus $k\text{NNSVM}_q(\mathbf{q}) = k\text{NNSVM}(\mathbf{q})$.

*k*NNSVM requires that the training of an SVM on the *k*-neighbourhood of the query point must be performed in the prediction step. This approach is computationally feasible only for problems with few points to test which is a condition that rarely holds in real-world classification problems. In general, we need to speed-up the prediction phase. The first modification of *k*NNSVM consists in predicting the label of a test point $\mathbf{q}$ using the local SVM model built on the *k*-neighbourhood of its nearest neighbour in $\mathcal{X}$. Formally, this can be written as:

$$k\text{NNSVM}_{\mathbf{t}}(\mathbf{q}) \text{ with } \mathbf{t} = \mathbf{x}_{r_{\mathbf{q}}(1)}. \tag{5}$$

Notice that in situations where the *k*-neighbourhood contains only one class the local model does not find any separation and so it can adopt the majority rule for improving the computational performances.

With this formulation the local learning can switch from the *lazy learning* (Aha, 1997) setting of the original formulation of *k*NNSVM to the *eager learning* setting with clear advantages in terms of prediction step complexity. This is possible computing a local SVM model for each $\mathbf{x} \in \mathcal{X}$ during the training phase obtaining the sets $\{(\mathbf{t}, k\text{NNSVM}_{\mathbf{t}}) \mid \mathbf{t} \in \mathcal{X}\}$ and applying the precomputed $k\text{NNSVM}_{\mathbf{t}}$ model such that $\mathbf{t} = \mathbf{x}_{r_{\mathbf{q}}(1)}$ for each query point $\mathbf{q}$ during the testing phase.

This approximation slightly modifies the approach of *k*NNSVM as a local learning algorithm. Instead of estimating the decision function for a *given* test example $\mathbf{q}$ and thus for a specific point in the input metric space, we estimate a decision function for *each* Voronoi cell $V_{\mathbf{x}}$ induced by the training set in the input metric space. In this way, the construction of the models in the training phase requires the estimation of N local decision functions. The prediction of a test point $\mathbf{q}$ is done using the model built for the Voronoi region in which $\mathbf{q}$ lies ($V_{\mathbf{h}}$ with $\mathbf{h} = \mathbf{x}_{r_{\mathbf{q}}(1)}$) that can be retrieved by searching for the nearest neighbour of $\mathbf{q}$ in $\mathcal{X}$.

4.2 Reducing the Number of Local Models that Need to Be Trained

The pre-computation of the local models during the training phase introduced above, increases the computational efficiency of the prediction step. However, a considerable overhead is added to the training phase. In fact, the training of an SVM for each training point can be slower than the training of a unique global SVM (especially for non small k values), so we introduce another modification of the method which aims to dramatically reduce the number of SVMs that need to be pre-computed. The idea is that we can relax the constraint that a query point $\mathbf{x}'$ is always evaluated using the model trained around its nearest training point. The decision function of this approach is

$$\mathsf{FastLSVM}(\mathbf{x}) = k\text{NNSVM}_{f(\mathbf{x})}(\mathbf{x}) \tag{6}$$

where $f : \mathcal{H} \mapsto \mathcal{C} \subseteq \mathcal{X}$ is a function mapping each unseen example $\mathbf{x}$ to a unique training example $f(\mathbf{x})$ which is, accordingly to Equation 4, the centre of the local model that is used to evaluate $\mathbf{x}$. The set $\mathcal{C}$ is the image of $f(\cdot)$, so $\mathcal{C} = f(\mathcal{H})$.

Notice that if $f(\cdot) = \mathbf{x}_{r_\cdot(1)}$, we have that $\mathcal{C} = \mathcal{X}$ and that $\mathsf{FastLSVM}(\mathbf{x})$ is equivalent to the kNNSVM formulation of Equation 5, and this can happen if we use *all* the examples in the training set as centres for local SVM models. In the general case, however, we select only a proper subset $\mathcal{C} \subset \mathcal{X}$ of points to be used as centres of kNNSVM models. In this case, if $\mathbf{x}_{r_\mathbf{x}(1)} \in \mathcal{C}$ then $f(\mathbf{x})$ can be defined as $f(\mathbf{x}) = \mathbf{x}_{r_\mathbf{x}(1)}$, but if $\mathbf{x}_{r_\mathbf{x}(1)} \notin \mathcal{C}$ then $f(x)$ must be defined in a way such that the principle of locality is preserved and the retrieval of the model is fast at prediction time.

Two aspects need to be addressed now: the strategy to select the subset $\mathcal{C}$ of $\mathcal{X}$, and the formulation of the function f associating each query example with an example in $\mathcal{C}$.

4.2.1 SELECTING THE CENTRES OF THE LOCAL MODELS

The approach we developed for selecting the set $\mathcal{C}$ of the centres of the local models is based on the idea that each training point must be in the k'-neighbourhood of at least one centre with k' being a fixed parameter and $k' \leq k$. From a slightly different viewpoint, we need to cover the entire training set with a set of hyper-spheres whose centres will be the examples in $\mathcal{C}$ and each hyper-sphere contains exactly k' points. We can formalise this idea with the concept of k'-neighbourhood covering set:

Definition 1 *Given $k' \in \mathbb{N}$, a k'-neighbourhood covering set of centres $\mathcal{C} \subseteq \mathcal{X}$ is a subset of the training set such that the following holds:*

$$\bigcup_{\mathbf{c} \in \mathcal{C}} \left\{ \mathbf{x}_{r_\mathbf{c}(i)} \mid i = 1, \ldots, k' \right\} = \mathcal{X}.$$

Definition 1 means that the union of the sets of the k'-nearest neighbours of $\mathcal{C}$ corresponds to the whole training set. Theoretically, for a fixed k', the minimisation of the number of local SVMs that we need to train can be obtained computing the SVMs centred on the points contained in the *minimal* k'-neighbourhood covering set of centres.

Definition 2 *The* Minimal k'-neighbourhood covering set of centres *is a k'-neighbourhood covering set $\mathcal{C} \subseteq \mathcal{X}$ which have the minimal cardinality.*

This problem is related to the *Set Cover Problem* (SC) (Garey and Johnson, 1979; Kearns and Vazirani, 1994; Marchand and Shawe-Taylor, 2003) and to the *Minimum Sphere Set Covering Problem* (MSSC) (Chen, 2005). However, in the SC and MSSC problems one specifies the radius of the spheres rather than their cardinality in terms of points they contain and it is not required that the centres of the hyperspheres correspond to points in the set. It is easy to show that MSSC is NP-hard but some efficient approximated results are available based on greedy approaches (Chvatal, 1979; Wang et al., 2006), integer and linear programming (Wei and Li, 2008).

In our case, however, we do not need the minimality of the constraints of the k'-neighbourhood covering set of centres to be strictly satisfied, because training some more local SVMs is acceptable instead of solving an NP-hard problem.

The heuristic procedure we developed can be seen as a modification of the greedy approach for the MSSC problem (Chvatal, 1979; Wang et al., 2006). The first k'-neighbourhood is selected randomly choosing its centre in $\mathcal{X}$, the following k'-neighbourhoods are retrieved selecting the centres that are still not members of other k'-neighbourhoods and are as far as possible from the already selected centres. The selection of the farthest example, still not included in the k'-neighbourhoods, as the centre of the next k'-neighbourhood, is the counterpart of the selection of the set of points having the minimum overlapping with the already covered set of points used by the greedy approach to the MSSC and SC problems.

For detailing the greedy approach we adopt, we need the concepts of minimum and maximum distance between the elements of a set of points A defined respectively as:

$$d(A) = \min \|\mathbf{x} - \mathbf{x}'\| \text{ with } \mathbf{x}, \mathbf{x}' \in A \text{ and } \mathbf{x} \neq \mathbf{x}'$$

and

$$D(A) = \max \|\mathbf{x} - \mathbf{x}'\| \text{ with } \mathbf{x}, \mathbf{x}' \in A.$$

In particular, the minimum distance between points in $\mathcal{X}$ is $m = d(\mathcal{X})$ and the maximum is $M = D(\mathcal{X})$. Our intention is to identify a system of subsets $S_i \subseteq \mathcal{X}$ with decreasing minimum distances $d(S_i)$; we can in this way define an ordering on the sets $\ldots \subset S_{i+1} \subset S_i \subset S_{i-1} \subset \ldots$ such that $\ldots > d(S_{i+1}) > d(S_i) > d(S_{i-1}) > \ldots$. With this strategy we can choose the centres of the local models first in the set S_{i+1}, then in the set S_i and so on, thus selecting first the centres that are assured to be distant at least $d(S_{i+1})$, then at least $d(S_i) < d(S_{i+1})$ and so on. More in detail, we require that in the ith set $S_i \subseteq \mathcal{X}$ the two nearest points are farther than b^i with $b > 1$, that is, they are subject to the constraint $d(S_i) > b^i$ with $b > 1$. The bound on the minimum distance $d(S_i)$ thus varies as powers of b depending on the set S_i.

Let us define precisely the system of sets $\{S_i\}$. The maximum i index of S_i is named top and the minimum is named bot, and they are univocally defined as those indexes satisfying $b^{top-1} \leq M < b^{top}$ and $b^{bot} < m \leq b^{bot+1}$. The S_i are recursively defined as:

$$\begin{cases} S_{top} &= \{\text{choose}(\mathcal{X})\} \\ S_i &= S_{i+1} \cup \underset{S \in \mathcal{X} \setminus S_{i+1}}{\text{argmax}} (|S| \text{ s.t. } d(S_{i+1} \cup S) > b^i) \quad \text{for } i = top-1, \ldots, bot \end{cases}, \tag{7}$$

where choose(A) is a function that selects only one element of the non-empty set A. An example of choose() for our case can be the following definition that selects the example with the minimum index:

$$\text{choose}(A) = \mathbf{x}_i \text{ with } i = min(z \in \mathbb{N} | \mathbf{x}_z \in A).$$

Notice that, since S_i contains S_{i+1} we have that

$$S_{top} = \{\text{choose}(\mathcal{X})\} \subseteq S_{top-1} \subseteq \ldots \subseteq S_{bot+1} \subseteq S_{bot} = \mathcal{X} \tag{8}$$

and, forcing for definition that $d(A) = \infty$ if $|A| = 1$,

$$d(S_{top}) = \infty > d(S_{top-1}) = M > d(S_{top-2}) > \ldots > d(S_{bot+1}) > d(S_{bot}) = m.$$

We can now formalise the selection of the centres from $\mathcal{X}$ using the S_i sets. The first centre $\mathbf{c}_1$ is simply the (only) example in S_{top}. The next centre $\mathbf{c}_2$ is chosen among the non-empty S_l sets obtained removing from S_i the first centre $\mathbf{c}_1$ and the points in its k'-neighbourhood; in particular $\mathbf{c}_2$ is chosen from the non-empty S_l with highest l. The general case for the $\mathbf{c}_j$ centre is similar, with the only difference being that we remove from the S_i sets all the centres $\mathbf{c}_t$ with $t < j$ and their k'-neighbourhood. More formally:

$$\begin{cases} \mathbf{c}_1 &= \text{choose}(S_{top}) \\ \mathbf{c}_j &= \text{choose}(S_l) \text{ with } l = \max(m \in \mathbb{N} | S_m \setminus \mathcal{X}_{\mathbf{c}_{j-1}} \neq \emptyset) \end{cases}, \tag{9}$$

where

$$\mathcal{X}_{\mathbf{c}_{j-1}} = \bigcup_{l=1}^{j} \left\{ \mathbf{x}_{r_{\mathbf{c}_l}(h)} \, | h = 1, \ldots, k' \right\}.$$

is the union of all the k'-neighbourhoods of the centres already included in $\mathcal{C}$.

We can briefly show that the $\mathcal{C}$ set found with Equation 9 is a k'-neighbourhood covering set of centres. In fact, the iterative procedure for selecting the centres in $\mathcal{C}$ terminates when the choose() function cannot select a point from S_l because all S_j with $j = bot, \ldots, top$ are empty. Since for the set S_{bot} we always have that $S_{bot} = \mathcal{X}$, this happens only when $\mathcal{X}_{\mathbf{c}_{i-1}} = \mathcal{X}$. Noticing that $\mathcal{X}_{\mathbf{c}_i}$ in this situation is equivalent to the constraint of Definition 1, we can conclude that $\mathcal{C}$ is a k'-neighbourhood covering set of centres.

Computationally, the selection of the centres from the S_j sets with Equation 9 can be performed efficiently once the S_j are identified. More problematic is the construction of the nested set of S_j sets. We can however notice that the S_j sets share some characteristics with the levels of cover trees. First, from Equation 7 we can easily see that for each S_j set with $j < top$ all the points in it are at least distant as b^j because $d(S_j) > b^j$; this is equivalent to the separation invariant of cover trees reported in Section 3.4. Second, always from Equation 7 we can conclude that each S_j is contained in every S_t set with $t < j$ as also explicated in Equation 8; this is equivalent to the nesting invariant of cover trees. The only constraint of our strategy to identify the S_j sets that is not respected by cover trees is the maximality of the set added to each S_j set to obtain S_{j+1}. However, the procedure to insert a new point in a cover tree is based on adding it to the highest possible level, and this is an efficient approximation of the maximality constraint we have in Equation 7. Taking all these facts into consideration, we chose to use the levels of cover tree as the S_j sets from which we select the centres as reported in Equation 9.

Consequently with the goal of reducing the number of local models, this approach no longer requires that a local SVM is trained for each training example, but we need to train only $|\mathcal{C}|$ SVMs centred on each $c \in \mathcal{C}$ obtaining the following models:

$$k\text{NNSVM}_{\mathbf{c}}(\mathbf{x}), \quad \forall \mathbf{c} \in \mathcal{C}.$$

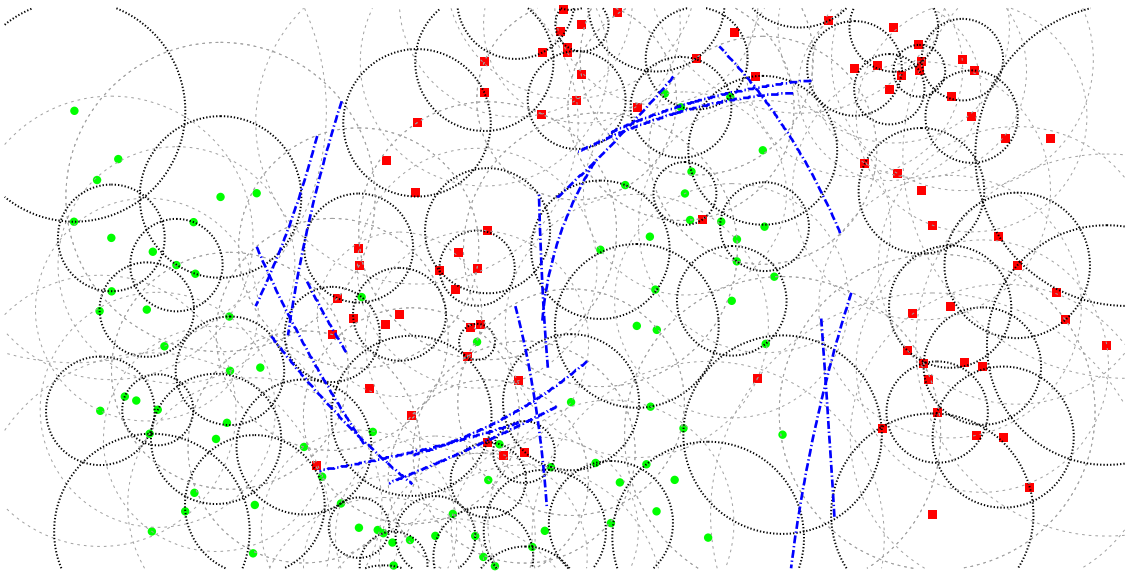


Figure 1: Graphical representation of the proposed approach using local models with $k' = 4$, $k = 15$, and local SVM with RBF kernel. The bold dotted circles highlights the k'-neighbourhoods covering all the training set (with some unavoidable redundancy), the thin dotted circles denotes the k-neighbourhoods on which the local models are trained. Some k-neighbourhoods do not produce an explicit decision function because entirely composed by points of the same class. The local SVM (with RBF kernel) decision functions are drawn in blue. Notice that, due both to the adoption of the k'-neighbourhood cover set and to the fact that only a fraction of the neighbourhoods need to be trained, we have only 17 local decision functions for 185 points.

Moreover if a neighbourhood contains only points belonging to one class the local model is the majority rule (specifically, unanimity) and the training of the SVM is avoided.

Figure 1 graphically shows the result of adopting the approach described above on a simple artificial data set with k and k' chosen for illustrative purposes. In fact, the example just aims to show the intuition behind the approach that is instead developed for large data sets and for non-extreme values of the neighbourhood parameters.

From Figure 1 we can also notice that the level of overlapping between k'-neighbourhoods and thus between k-neighbourhoods depends on the value of k'. If k' is low, a large number of k'-neighbourhoods are required to cover the entire training set, whereas if k' is large fewer k'-neighbourhoods are needed. The k' parameter thus tune the level of redundancy of the local models.

4.2.2 SELECTING THE LOCAL MODELS FOR TESTING POINTS

Once the set of centres $\mathcal{C}$ is defined and the corresponding local models are trained, we need to select the proper model to use for predicting the label of a test point. A simple strategy we can adopt consists in selecting the model whose centre $\mathbf{c} \in \mathcal{C}$ is the nearest centre with respect to the

testing example. Using the general definition of FastLSVM of Equation 6 with $f(x) = r^{\mathcal{C}}_{\mathbf{x}}(1)$ where $r^{\mathcal{C}}$ corresponds to the reordering function defined in Equation 3 performed on the $\mathcal{C}$ set instead of $\mathcal{X}$, the method, called FaLK-SVMc, is defined as:

$$\text{FaLK-SVMc}(\mathbf{x}) = k\text{NNSVM}_{\mathbf{c}}(\mathbf{x}) \text{ where } \mathbf{c} = \mathbf{x}_{r^{\mathcal{C}}_{\mathbf{x}}(1)}. \tag{10}$$

FaLK-SVMc is satisfactory from the computational viewpoint, for it performs the nearest neighbour search on $\mathcal{C}$ only. However, it does not assure that the testing point is evaluated with the model centred on the point for which the testing point itself is the nearest in terms of neighbour ranking. For example, a testing point $\mathbf{q}$ can be closer to $\mathbf{c}_1$ than $\mathbf{c}_2$ using the Euclidean distance, but at the same time we can have that $\mathbf{q}$ is the i-th nearest neighbour of $\mathbf{c}_1$ in $\mathcal{X}$ and the j-th nearest neighbour of $\mathbf{c}_2$ with $i > j$. This is a problem because using the model centred on $\mathbf{c}_2$ is better in terms of proximity. In order to overcome this issue of FaLK-SVMc we propose to use, for a testing point $\mathbf{q}$, the model centred on the training point which is the nearest in terms of the neighbourhood ranking to its training nearest neighbour. We can do this defining a function $cnt : \mathcal{X} \mapsto \mathcal{C}$ in the following way:

$$\begin{aligned} cnt(\mathbf{x}_i) &= \text{choose}(\left\{\mathbf{c}_z \in \mathcal{C} | \mathbf{x}_i = \mathbf{x}_{r_{\mathbf{c}_z}(h)}\right\}) \\ \text{where } h &= \min\left(t \in \{1,\ldots,k'\} \middle| \mathbf{x}_{r_{\mathbf{c}_j}(t)} = \mathbf{x}_i \text{ and } \mathbf{c}_j \in \mathcal{C}\right). \end{aligned} \tag{11}$$

The cnt function finds, for each example $\mathbf{x}$, the minimum value h such that $\mathbf{x}$ is in the h-neighbourhood of at least one centre $\mathbf{c} \in \mathcal{C}$; then, among the centres having $\mathbf{x}$ in their h-neighbourhoods, it selects the centre with the minimum index. The existence of h is guaranteed by the k'-neighbourhood covering strategy. In this way each training point is univocally assigned to a centre and so the decision function of this approximation of Local SVM derivable from FastLSVM of Equation 6 with $f(\mathbf{x}) = cnt(\mathbf{x})$, and called FaLK-SVM, is simply:

$$\text{FaLK-SVM}(\mathbf{x}) = k\text{NNSVM}_{cnt(\mathbf{t})}(\mathbf{x}) \text{ where } \mathbf{t} = \mathbf{x}_{r_{\mathbf{x}}(1)}. \tag{12}$$

The association between training points and centres defined by Equation 11 can be efficiently precomputed during the training phase, delaying to the testing phase only the retrieval of the nearest neighbour of the testing point and the evaluation of the local SVM model.

Figure 2 graphically shows the application of the FaLK-SVM($\mathbf{x}$) prediction strategy on a toy data set; the training phase for the same data set is illustrated in Figure 1.

FaLK-SVM can be generalised for multi-class problems in the same way of kNNSVM, but in this paper we focus on binary problems in order to better evaluate the approach.

4.3 FaLK-SVM with Internal Model Selection: FaLK-SVMI

For training a kernel machine, once a proper kernel is chosen, it is crucial to carefully tune the kernel parameters and, for SVM, to set the soft margin regularisation constant C. Model selection is very often performed estimating the empirical error with different parameter values and a popular method is the κ-fold cross-validation[2] with a grid search on parameter space. Given the following loss function for the two-class classification case

$$L(y, \text{SVM}(\mathbf{x})) = \begin{cases} 0 & \text{if } y = \text{SVM}(\mathbf{x}) \\ 1, & \text{if } y \neq \text{SVM}(\mathbf{x}) \end{cases},$$

2. Although κ can be confused with the neighbourhood size k or with the kernel function K, κ is always used for denoting κ-fold CV, so the context should be sufficient to avoid ambiguity.

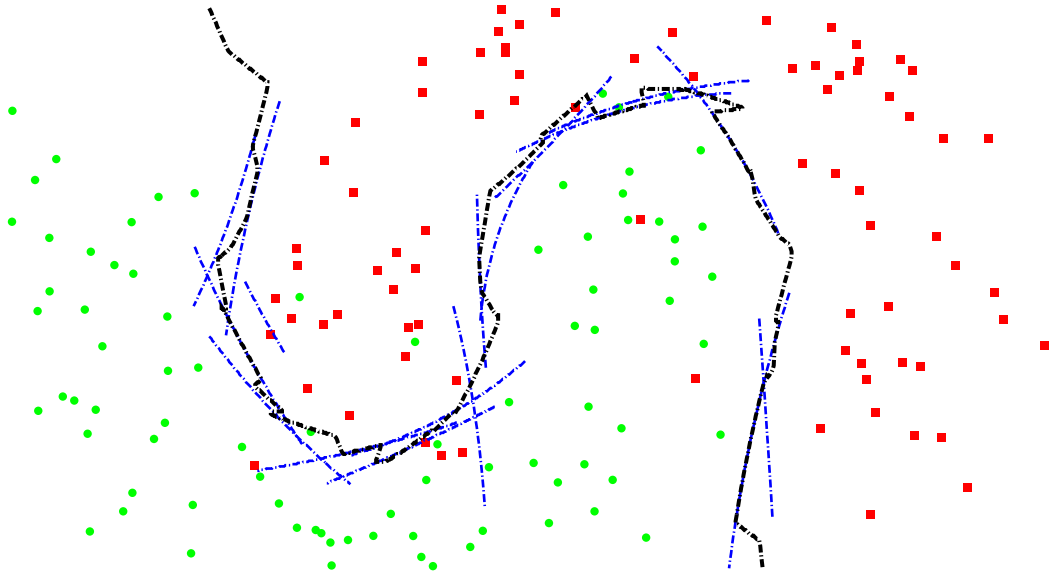


Figure 2: Graphical representation of the global decision function (black dotted line) obtained with the local decision functions (the same of Figure 1) using the described approach that uses for each query point the local decision function of the Voronoi region in which it lies.

and partitioning the training set $\mathcal{X}$ in κ subsets each with the same cardinality[3] (called folds), the κ-fold cross validation (CV) procedure consists in searching for the parameters that minimise the average of the losses on $\mathcal{X}_f$ of the classifier trained on $\mathcal{X} \setminus \mathcal{X}_f$ for $f = 1, \ldots, \kappa$. The effectiveness in terms of testing accuracies of κ-fold CV is high, but it adds a computational overhead to the training phase. In fact, the computational complexity of a κ-fold CV run on a single parameter choice is in the order of κ times the training time; if we have p parameters to set and c possible choices for each parameter, the κ-fold cross-validation with grid selection is $\kappa \cdot c^p$ times slower than a single training of the classifier.

The model selection for FaLK-SVM and FaLK-SVMc can be performed using κ-fold CV. The only difference with SVM is that our local kernel machines need to estimate an additional parameter which is the neighbourhood size k (which is however usually chosen in a small set of possible values). However, with the local setting of the classification problem we are discussing in this paper, it is also possible to efficiently tackle the complexity of the model selection phase. Basically, since FaLK-SVM trains a set of local models, we can perform the model selection in a grid-search setting on a subset of the neighbourhoods. In this way we can efficiently estimate the global parameters of FaLK-SVM without considering all the training points during model selection. The classifier implementing this approach to model selection is called FaLK-SVMl.

As a first step for defining the model selection approach of FaLK-SVMl, we define a different setting of model selection for kNNSVM.

3. Without loss of generality, we assume $|\mathcal{X}| \mod \kappa = 0$.

Definition 3 (Localised κ-fold CV model selection for *k*NNSVM) *The procedure applies the κ-fold CV model selection on the k-neighbourhood of the query point.*

However, since the local model is used by *k*NNSVM only for the central point, the model selection should be performed in order to make the local models predictive especially for the very internal points. The idea thus consists in selecting the κ validation sets exclusively from the k' most internal points, taking as each corresponding training fold the union of the remaining k'-neighbourhood points and of the $k-k'$ most external points of the k-neighbourhood.

Definition 4 (k'-internal κ-fold CV model selection for *k*NNSVM)
The procedure applies the localised κ-fold CV model selection on the k'-neighbourhood of the query point in the training set adding to each training fold the points in the k-neighbourhood that are not in the k'-neighbourhood with $k > k'$.

For FaLK-SVM we can apply the k'-internal κ-fold CV for *k*NNSVM model selection on a randomly chosen training example and use the resulting parameters for all the local models. In order to be robust the procedure is repeated on more than one k-neighbourhood choosing the parameters that minimise the average k'-internal κ-fold CV error among the k-neighbourhoods.

Definition 5 (k'-internal κ-fold CV model selection for FaLK-SVM)
The procedure applies the k'-internal κ-fold CV for kNNSVM model selection on the k-neighbourhoods of $1 \leq m \leq |\mathcal{C}|$ randomly chosen centres selecting the parameters that minimise the average error rate among the m applications.

The variant of FaLK-SVM that adopts the k'-internal κ-fold CV described in Definition 5 is named FaLK-SVMI. Since FaLK-SVMI selects the local model parameters using a small subset of the training set, the variance of the error may be higher than the standard cross-validation strategies. However, for huge data sets the standard model selection can be too slow to be applied and, in any case, one may use large values of m to decrease the risk of selecting non-optimal parameters.

4.3.1 A SPECIFIC STRATEGY FOR SETTING THE RBF KERNEL WIDTH

As already proposed by Tsang et al. (2005) and by Segata and Blanzieri (2009b), good choices for the RBF kernel width σ of SVM are based on the median (or other percentiles) of the distribution of distances. In FaLK-SVMI we can thus efficiently estimate σ for each local model simply calculating the median of the distances in the neighbourhood. This approach has some analogies with standard SVM using a variable RBF kernel width that have good potentialities for classification (Chang et al., 2005). Since other percentiles different from the median can give better accuracy performances, in FaLK-SVMI the percentile can be a value to set using the k'-internal κ-fold CV approach.

4.4 Generalisation Bounds for *k*NNSVM and FaLK-SVM

The class of LLAs introduced by Bottou and Vapnik (1992) includes *k*NNSVM, and can be theoretically analysed using the framework based on the local risk minimisation (Vapnik and Bottou, 1993; Vapnik, 2000). On the other hand, FaLK-SVM is not a LLA as intended by Bottou and Vapnik (1992). In fact, LLAs compute the local function for each specific testing point thus delaying the neighbourhood retrieval and model training until the testing point is available. However, we show here that generalisation bounds for FaLK-SVM can be derived starting from the LLA ones.

We need to recall the bound for the local risk minimisation, which is a generalisation of the global risk minimisation theory.

Theorem 6 (Vapnik (2000)) *For a testing point* $\mathbf{x}'$ *and with probability* $1-\eta$ *simultaneously for all bounded functions* $A \leq L(y, f(\mathbf{x},\alpha)) \leq B$, $\alpha \in \Lambda$ *(where* Λ *is a set of parameters), and all locality functions* $0 \leq T(\mathbf{x},\mathbf{x}_0,\beta) \leq 1$, $\beta \in (0,\infty)$, *the following inequality holds true:*

$$R^{LLA}(\alpha,\beta,\mathbf{x}') \leq \frac{\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i,\alpha))T(\mathbf{x}_i,\mathbf{x}',\beta) + (B-A)\gamma(N,h^{\Sigma})}{|\frac{1}{N}\sum_{i=1}^{N} T(\mathbf{x}_i,\mathbf{x}',\beta) - \gamma(N,h^{\beta})|},$$

where

$$\gamma(N,h) = \sqrt{\frac{h\ln(2N/h+1) - \ln\eta/2}{N}},$$

and h^{Σ} *is the VC dimension of the set of functions* $L(y_i, f(\mathbf{x}_i,\alpha))T(\mathbf{x}_i,\mathbf{x}',\beta), \alpha \in \Lambda, \beta \in (0,\infty)$ *and* h^{β} *is the VC dimension of* $T(\mathbf{x}_i,\mathbf{x}',\beta)$

For kNNSVM, the loss function is simply

$$L(y_i, f(\mathbf{x}_i,\alpha)) = \begin{cases} 0 & \text{if } y_i = f(\mathbf{x}_i,\alpha) \\ 1 & \text{if } y_i \neq f(\mathbf{x}_i,\alpha) \end{cases}$$

and the locality function is

$$T(\mathbf{x}_i,\mathbf{x}',k) = \begin{cases} 1 & \text{if } \exists j \leq k \text{ s.t. } i = r_{\mathbf{x}'}(j) \\ 0 & \text{otherwise} \end{cases}.$$

It is straightforward to show that $\sum_{i=1}^{N} T(\mathbf{x}_i,\mathbf{x}',k) = k$. Moreover $T(\mathbf{x}_i,\mathbf{x}',k)$ has VC dimension equal to 2; it is, in fact, the class of functions corresponding to hyperspheres centred on $\mathbf{x}'$ with diameters equal to the distances of the points from $\mathbf{x}'$ and can thus shatter any set of two points with different classes, but cannot shatter three points with the nearest and furthest points having a class different from the third point.

We observe that, in our case,

$$\sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i,\alpha))T(\mathbf{x}_i,\mathbf{x}',\beta) = \sum_{i=1}^{k} L(y_i, f(\mathbf{x}_i,\alpha))$$

and so we can obtain:

$$R^{k\text{NNSVM}}(\alpha,k,\mathbf{x}') \leq \frac{\frac{1}{N}k\cdot\nu_{\mathbf{x}'} + \gamma(N,h^{\Sigma})}{|\frac{1}{N}k - \gamma(N,2)|} \tag{13}$$

where $\nu_{\mathbf{x}'}$ is the ratio of misclassified training points in the k-neighbourhood of $\mathbf{x}'$.

The possibility of local approaches to obtain a lower bound on test misclassification probability acting with the locality parameter, as stated in Vapnik and Bottou (1993); Vapnik (2000) for LLA, it is even more evident for kNNSVM considering Equation 13. In fact, although choosing a $k < N$ is not sufficient to lower the bound, as the model training becomes more and more local k decreases and (very likely) the misclassification training rate $\nu_{\mathbf{x}'}$ decreases as well. Moreover, also the complexity of the classifier (and thus h^{Σ}) can decrease when the neighbourhood decreases, because simpler decision functions can be used when fewer points are considered. Taking this into

consideration, it is necessary to consider the trade-off between the degree of locality k, the function of the empirical error with respect to k and the complexity of the local classifier needed with respect to k, in order to find a minimum of the expected risk which is lower than the $k = N$ case. Multiple strategies can be used to tune this trade-off, especially if prior or high-level information are available for a specific problem; since in this work we aim to be as general as possible, the expected risk is estimated for the computational experiments using cross-validation based approaches.

FaLK-SVM pre-computes local models to be used for testing points lying in sub-regions (k-NN Voronoi cells) of the training set. The risk associated to FaLK-SVM considering a specific query point $\mathbf{x}'$ can be defined using the risk of kNNSVM, supposing that $\mathbf{x}' \in V_{\mathbf{x}_i}$ and so $\mathbf{x}_{r_{\mathbf{x}'}(1)} = \mathbf{x}_i$:

$$R^{\text{FaLK-SVM}}(\alpha, k, \mathbf{x}') = R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}') + \lambda(\mathbf{x}', \mathbf{x}_{r_{\mathbf{x}'}(1)}) \leq R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}') + \lambda_{r_{\mathbf{x}'}(1)} \tag{14}$$

where $\lambda(\mathbf{x}', \mathbf{x}_{r_{\mathbf{x}'}(1)})$ is due to the approximation introduced, for the prediction of the label of the query point $\mathbf{x}$', by the use of the k-neighbourhood of $r_{\mathbf{x}'}(1)$ instead of the k-neighbourhood of $\mathbf{x}'$ itself and

$$\lambda_{r_{\mathbf{x}'}(1)} = \max_{x'' \in V_{\mathbf{x}_i}} \lambda(\mathbf{x}'', \mathbf{x}_{r_{\mathbf{x}'}(1)}).$$

If we consider $k' = 1$, the approximation is due to the fact that $\{r_{\mathbf{c}}(i) | \, i = 1, \ldots, k\}$ and $\{r_{\mathbf{x}'}(i) | \, i = 1, \ldots, k\}$ can be slightly different; however, considering a non very low value for k, the differences between the two sets are possible only for the very peripheral points of the neighbourhoods which are those that influence less the shape of the decision function in the central region. We will empirically show that $\lambda_{r_{\mathbf{x}'}(1)}$ is, on average, a small penalising term that still permits to achieve lower risks than SVM using k' values higher than 1.

The risk of FaLK-SVM in its eager learning setting (i.e., without the explicit dependency on the query point) can thus be defined as:

$$\begin{aligned} R^{\text{FaLK-SVM}}(\alpha, k) &= \int_{\mathbf{x}'} R^{\text{FaLK-SVM}}(\alpha, k, \mathbf{x}') g(\mathbf{x}') d\mathbf{x}' \\ &\leq \int_{\mathbf{x}'} \left(R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}_i) + \lambda_{r_{\mathbf{x}'}(1)} \right) g(\mathbf{x}') d\mathbf{x}' \\ &= \int_{\mathbf{x}'} R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}_i) g(\mathbf{x}') d\mathbf{x}' + \int_{\mathbf{x}'} \lambda_{r_{\mathbf{x}'}(1)} g(\mathbf{x}') d\mathbf{x}' \\ &= \int_{\mathbf{x}'} R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}_i) g(\mathbf{x}') d\mathbf{x}' + E[\lambda]. \end{aligned} \tag{15}$$

where $E[\lambda]$ is the expectation of the term due to the use of the kNNSVM risk for FaLK-SVM as discussed above.

4.5 Computational Complexity Analysis

We analyse here the computational performances of FaLK-SVM from the theoretical complexity viewpoint. The training phase of FaLK-SVM can be subdivided in four steps:

- the building of the cover tree that scales as $O(N \log N)$;
- the retrieval of the local models that scales as $O(|\mathcal{C}| \cdot k \log N)$;
- the assignment of each point to a k'-neighbourhood that scales as $O(N)$;

- the training of the local SVM models that scales as $O(|\mathcal{C}| \cdot k^3)$.

The overall training time, considering the worst case in which $k' = 1$ so $|\mathcal{C}| = N$, scales as:

$$O(N \log N + \mathcal{C} \cdot k \log N + N + \mathcal{C} \cdot k^3) = O(kN \cdot \max(\log N, k^2))$$

which is, considering a reasonably low and fixed value for k as happens in practice for large data sets, sub-quadratic, and in particular $O(N \log N)$, in the number of training points.

For the testing phase of FaLK-SVM we can distinguish two steps (for each testing point):

- the retrieval of the nearest training point that scales as $O(\log N)$;
- the prediction of the testing label using the selected local model that scales as $O(k)$.

The testing can thus be performed in $O(\max(\log N, k))$, so it is logarithmic in N. FaLK-SVMc is even faster because it scales as $O(\max(\log |\mathcal{C}|, k)) \leq O(\max(\log N, k))$.

FaLK-SVM is thus asymptotically faster than SVM (also considering the worst case in which SVM scales quadratically and $k' = 1$) and all the classifiers taking more than $O(N \log N)$ for training and $O(\log N)$ for testing. Moreover, FaLK-SVM can be very easily parallelised differently from SVM whose parallelisation, although possible (Zanni et al., 2006; Dong, 2005), is rather critical; for FaLK-SVM is sufficient that, every time the points for a model are retrieved, the training of the local SVM is performed on a different processor. In this way the time complexity of FaLK-SVM can be further lowered to $O(N \cdot \max(k \log N, k^3 / N_{proc}))$ where N_{proc} is the number of processors.

Another advantage of FaLK-SVM over SVM is space complexity. Since FaLK-SVM performs SVM training on small subregions (assuming a reasonable low k), there are no problems of fitting the kernel matrix into main memory. The overall required space is, in fact, $O(N + k^2)$, that is, linear in N, which is much lower than SVM space complexity of $O(N^2)$. For large data sets, FaLK-SVM can still maintain in memory the entire local kernel matrix (if k is not too large), whereas SVM must discard some kernel values thus increasing SVM time complexity due to the need of recomputing them. Analysing the space required to store the trained model in secondary storage devices (e.g., hard disks), we can notice that FaLK-SVM needs to save in the model file the entire set of local models; although we store the models with pointers to the training set points, we need to maintain the whole training set in the model file (or give as input for the testing module both the model file and the original training set). FaLK-SVM, in other words, needs to store the training set also in the model file, differently from SVM that needs to store only the support vectors (whose number however grows linearly with N).

4.5.1 CURSE OF DIMENSIONALITY

Although not explicitly considered here, cover trees have a constant in the complexity bounds depending on the so-called doubling constant (Clarkson, 1997; Krauthgamer and Lee, 2004) which is a robust estimation of the intrinsic dimensionality of the data. Notice that the intrinsic dimensionality of a data set can be much lower than the dimensionality intended simply as the number of features. Regardless of the doubling constant, FaLK-SVM maintains the derived complexity bounds[4] with respect to N, but the overhead introduced for building the cover tree and retrieving the k-neighbourhoods can be very high. This drawback, due to the well-known problem of the *curse of*

4. The high intrinsic dimensionality can cause the need for an high value of $|\mathcal{C}|$, but in the bound we already considered the worst case in which $k' = 1$ and thus $|\mathcal{C}| = N$.

Algorithm 1 FaLK-SVM-train (training set **x[]**, training size **n**, neighbourhood size **k**, assignment neighbourhood size **k'**)

```
models[] ⇐ null   // the set of models
modelPtrs[] ⇐ null   // the set of pointers to the models
c ⇐ 0   // the counter for the centres of the models
indexes[] ⇐ {1,...,N}   // the indexes for centres selection
Randomise indexes   // randomise the indexes
for i ⇐ 1 to N do
    index ⇐ indexes[i]   // get the i-th index
    if modelPtrs[index] = null then   // if the point has not been assigned to a model...
        localPoints[] ⇐ get ordered kNN of x[i]   // ...retrieve its k-neighbourhood ...
        models[c] ⇐ SVMtrain on localPoints[]   // ...train a local SVM...
        modelPtrs[index] ⇐ models[c]   // ...assign the centre to the trained model.
        for j = 1 to k' do   // Assign the model to the k'<k nearest neighbours of the centre
            ind ⇐ get index of localPoints[j]
            if modelPtrs[ind] = null then   // assign the points in the k'-neighbourhood ...
                modelPtrs[ind] ⇐ models[c]   // ...to the c-th model
            end if
        end for
        c ⇐ c+1
    end if
end for
return models, modelPtrs
```

Algorithm 2 FaLK-SVM-predict (training set **x[]**, points-to-model pointers **modelPtrs**, Local SVM models **models**, query point **q**)

```
Set p = get NN of q in x   // retrieve the nearest training point with respect to q...
Set nnIndex = get index of p   // ...retrieve its index ...
return label = SVMpredict q with modelPtrs[nnIndex]   // ...and use the corresponding model
for predict the label of the query point.
```

dimensionality that affects also SVM with local kernels (Bengio et al., 2005), is not however crucial here, as we are considering non-linear classification problems that are not high-dimensional. In fact, apart from computational problems, high-dimensional problems are typically tackled by approaches not related with the concept of locality (e.g., linear SVM instead of SVM with a RBF kernel).

4.6 Implementation and Availability

FaLK-SVM (and also FkNN and FkNNSVM that are the implementations of kNN and *k*NNSVM using cover trees) is available as part of the Fast Local Kernel Machine Library (Segata, 2009, FaLKM-lib). FaLK-SVM is written in *C*/*C*++ and it uses LibSVM v. 2.88 (Chang and Lin, 2001) for local SVM training and testing whereas we use our own implementation of the cover trees data-structure. The pseudo-code for the training phase is reported in Algorithm 1 and for the testing phase

method	brief description
FkNN	implementation of kNN (Section 3.1) with cover trees
FkNNSVM	implementation of *k*NNSVM (Section 3.3) with cover trees
FaLK-SVM	implementation of fast and scalable local kernel machines (see Equation 12)
FaLK-SVM-train	module for the training of FaLK-SVM (see Algorithm 1)
FaLK-SVM-predict	module for the testing of FaLK-SVM (see Algorithm 2)
FaLK-SVMc	faster prediction variant of FaLK-SVM (see Equation 10)
FaLK-SVMl	implementation of FaLK-SVM with local model selection (Section 4.3)
FkNNSVM-nr	implementation of *k*NNSVM for noise reduction (Segata et al., 2009b)
FaLKNR	impl. of noise reduction with FaLK-SVM (Segata et al., 2009a)

Table 1: Summary for the classifiers developed in the local kernel machine framework and implemented in FaLKM-lib.

in Algorithm 2 (use of cover trees and minimisation of t in Equation 11 are omitted for clearness). Table 1 summarizes the classifiers discussed in this paper and the modules of FaLKM-lib.

5. Empirical Analysis

The empirical analysis is organised into three experiments performed with different objectives and using different data sets. Experiment 1 (Section 5.1) has the objective of assessing the generalisation performances of FaLK-SVM with respect to SVM (using LibSVM) and to *k*NNSVM (using FkNNSVM) and thus assessing if FaLK-SVM is more accurate than SVM and if it is a good approximation of *k*NNSVM. For this experiment we use 25 non-large data sets. Experiment 2 (Section 5.2) focuses on comparing the classification accuracies and the computational performances of FaLK-SVM (and its variants FaLK-SVMc and FaLK-SVMl) with respect to SVM (using LibSVM) on large data sets. For this experiment we use 8 data sets with training set cardinalities ranging from about 50k examples to more than 1 million. Experiment 3 (Section 5.3) aims to understand (i) whether FaLK-SVM has better scalability and accuracy performances than LibSVM, a number of approximated SVM solvers (CVM, BVM, LASVM, CPSP and USVM) and SVM-bagging and (ii) which are the computational and accuracy differences between FaLK-SVM, FaLK-SVMc and FaLK-SVMl. For this last experiment we use 4 data sets with increasing training set size up to 3 million examples. The experiments, unless otherwise specified, are carried out on an AMD Athlon 64 X2 Dual Core Processor 5000+, 2600MHz, with 3.56Gb of RAM with Linux operating system.

5.1 Experiment 1: Comparison of FaLK-SVM with LibSVM and FkNNSVM

In this evaluation we compare SVM (using LibSVM), *k*NNSVM (using FkNNSVM) and FaLK-SVM on 25 non-large data sets, with the objective of studying the generalisation performances of *k*NNSVM with respect to SVM and the level of approximation introduced by FaLK-SVM to the FkNNSVM algorithm.

data set name	# of features	# of points	class balancing	data set name	# of features	# of points	class balancing
sonar	60	208	53%/47%	fourclass	2	862	64%/36%
heart	13	270	56%/44%	tic-tac-toe	9	958	65%/35%
mushrooms	112	300	53%/47%	mam	5	961	54%/46%
haberman	3	306	74%/26%	numer	24	1000	70%/30%
liver	6	345	58%/42%	splice	60	1000	52%/48%
ionosphere	34	351	64%/36%	spambase	57	1000	57%/43%
vote	15	435	61%/39%	vehicle	21	1243	76%/24%
musk1	166	476	57%/43%	cmc	7	1473	57%/43%
hill-valley	100	606	51%/49%	ijcnn1	22	1500	68%/32%
breast	10	683	65%/35%	a1a	123	1605	76%/24%
australian	14	690	56%/44%	chess	35	2130	52%/48%
transfusion	4	748	76%/24%	astro	4	3089	65%/35%
diabetes	8	768	65%/35%				

Table 2: The 25 binary-class data sets of Experiment 1.

5.1.1 EXPERIMENTAL PROTOCOL

The data sets are listed in Table 2; they are retrieved from the UCI (Asuncion and Newman, 2007) and STATLOG (Michie et al., 1994) repositories, with cardinality between 200 and 3100 points (some data sets have been randomly sub-sampled), dimensionality lower than 200, not very unbalanced, and they are all scaled in the $[0,1]$ interval. The comparison is carried out using three different kernel functions (linear, RBF and homogeneous polynomial), in a 10-fold CV experimental setting. Internal to each training fold the model selection is performed with a nested 10-fold CV choosing the parameters in the following ranges. The regularisation parameter C is chosen for all methods in the set $\{2^{-2}, 2^{-1}, \ldots, 2^{9}, 2^{10}\}$, the width parameter σ of the RBF kernel in $\{2^{-5}, 2^{-4}, \ldots, 2^{2}, 2^{3}\}$, the degree of the polynomial kernels in $\{1,2,3\}$. The neighbourhood parameter k for FkNNSVM and FaLK-SVM is selected by the cross-validation procedure in the set $\{2^1, 2^2, \ldots, 2^9, 2^{10}, |\mathcal{X}|\}$ where $|\mathcal{X}|$ is the cardinality of the training set,[5] while the k' parameter of FaLK-SVM is fixed to $k/2$ which is a value that privileges scalability over accuracy because we want to test a value that can permit good computational results for large and very large data sets.

5.1.2 RESULTS AND DISCUSSION

Table 3 reports the accuracy results of all tested methods and kernels. In addition to the mean ranks reported in the figure, we assessed the statistical significance of the differences between pairs of methods using the Wilcoxon Signed Rank Test (Wilcoxon, 1945; Demšar, 2006) with $\alpha = 0.05$. The test highlights that FkNNSVM is significantly better than LibSVM for the linear and polynomial kernels, whereas for the RBF kernel no significant differences are detected, although the mean rank of FkNNSVM with RBF kernel is lower than LibSVM with RBF kernel. Applied to FaLK-SVM, the Wilcoxon Signed Rank Test detects a significant difference with respect to LibSVM only for the linear kernel. If we perform the Friedman test (Friedman, 1940) ($\alpha = 0.05$), the null hypothesis is

5. For data set with less than 1024 points some k values are of course not tested.

data set	LibSVM			FkNNSVM			FaLK-SVM		
	K^{lin}	K^{rbf}	K^{hpol}	K^{lin}	K^{rbf}	K^{hpol}	K^{lin}	K^{rbf}	K^{hpol}
sonar	74.52	87.83	83.16	**89.36**	86.90	87.40	84.55	87.88	84.05
heart	**84.81**	82.22	**84.81**	**84.81**	81.11	**84.81**	83.70	81.85	83.70
mushrooms	97.99	98.33	98.32	98.67	98.33	98.6	**99.00**	**99.00**	**99.00**
haberman	73.20	73.20	72.89	**75.82**	75.16	74.18	73.25	73.20	73.87
liver	68.71	**74.24**	71.90	73.64	73.96	73.94	70.73	71.92	71.92
ionosphere	88.04	93.72	88.88	93.75	**94.59**	93.75	86.91	94.01	89.18
vote	94.95	96.32	94.95	96.32	**96.33**	96.32	94.94	96.32	94.94
musk1	86.55	94.54	93.07	89.44	**94.96**	91.17	87.18	93.90	92.43
hill-valley	63.70	**66.00**	63.70	64.86	65.18	64.86	65.17	64.03	65.00
breast	**96.78**	**96.78**	**96.78**	96.49	96.49	96.35	96.19	96.49	96.19
australian	85.50	84.78	84.20	84.78	**85.50**	84.92	85.07	85.07	84.78
transfusion	76.21	77.40	76.47	79.81	78.74	79.81	79.67	78.87	**79.94**
diabetes	76.54	76.54	76.68	76.81	**78.24**	77.07	75.90	76.68	75.12
fourclass	77.39	**100.00**	78.66	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
tic-tac-toe	98.33	99.68	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
mam	82.10	82.63	81.27	**82.95**	82.73	82.85	81.80	82.63	80.97
numer	**77.00**	75.90	76.50	76.30	75.70	76.00	76.70	74.70	75.90
splice	80.41	**86.70**	86.60	80.41	86.30	86.60	78.30	86.20	86.60
spambase	89.80	90.60	89.80	90.60	90.50	90.60	**90.70**	90.60	**90.70**
vehicle	82.71	84.16	84.80	82.78	84.64	84.71	83.27	84.72	**85.04**
cmc	59.26	65.45	64.16	62.46	**67.72**	63.61	63.61	65.31	64.36
ijcnn1	85.53	93.94	92.73	93.93	93.47	93.60	92.80	**94.47**	93.20
a1a	**83.43**	81.94	**83.43**	82.87	82.06	82.87	82.87	82.06	82.87
chess	96.57	98.45	98.03	97.84	**98.50**	98.08	97.32	98.45	98.08
astro	95.34	96.73	96.89	96.96	96.92	**97.05**	96.96	96.67	96.86
mean rank	7.04	4.60	5.80	4.38	3.86	4.02	5.72	4.56	5.02

Table 3: 10-fold CV accuracy results for the 25 data set of Experiment 1. The best results for each data set are highlighted in bold (taking into account all decimal values).

violated, but, according to the Nemenyi post-hoc test (Nemenyi, 1963) ($\alpha = 0.05$) the only method that is statistically significantly different from the others is SVM with linear kernel.

The observation that FkNNSVM is significantly better than SVM if a non-local kernel is used, is a confirmation of what we already noticed (Segata and Blanzieri, 2009a). Using the RBF kernel, instead, no significant differences are detected, although the mean rank of FkNNSVM with RBF kernel is lower than LibSVM with RBF kernel. This is mainly due to the fact that SVM with RBF kernel is already very accurate and significant improvements over it are very difficult. We may also say that locality is already included in the RBF kernel and thus, at least for non-large data sets, the adoption of a local method is somehow equivalent. Regarding FaLK-SVM, significant differences with respect to LibSVM are detected only for the linear kernel. Although FaLK-SVM does not achieve the accuracy results of FkNNSVM, if we look to the mean ranks, we can conclude that the approximation on the *k*NNSVM approach introduced in FaLK-SVM still permits to achieve slightly

data set name	# of feat.	train. points	testing points	class balancing	original source
ijcnn1	22	49990	91701	90%/10%	LibSVM rep. (Chang and Lin, 2001)
cov-type *	54	100000	481010	51%/49%	LibSVM rep. (Chang and Lin, 2001)
census-inc	41	199523	99762	94%/6%	UCI rep. (Asuncion and Newman, 2007)
cod-rna	8	364651	121549	67%/33%	(Uzilov et al., 2006)
intr-det	40	1026588	311029	79%/21%	UCI KDD rep. (Hettich and Bay, 1999)
2-spirals *	2	100000	100000	50%/50%	Synthetic (Segata and Blanzieri, 2009c)
ndcc *	5	100000	100000	61%/39%	Synthetic (Thompson, 2006)
checker-b *	2	300000	100000	50%/50%	Synthetic (e.g., see Tsang et al., 2005)

Table 4: The 8 large data sets of the second empirical experiment. The data sets whose extensions are used also in Experiment 3 are denoted with *.

better results than SVM also on non-large data sets, confirming our preliminary analysis (Segata and Blanzieri, 2009c). These results also indicates that the $E[\lambda]$ term introduced in the risk of FaLK-SVM (Section 4.4), due to the approximations introduced to the kNNSVM approach, is small enough to assure higher generalisation accuracies with respect to SVM.

The overall outcome of this experiment is that FaLK-SVM is a good approximation of FkNNSVM that maintains a little advantage over SVM and it is particularly effective with the RBF kernel with respect to linear and polynomial kernels. Notice that the experiment is carried out using small data sets in which locality is very likely to play a marginal role differently from large data sets in which it can be crucial.

5.2 Experiment 2: FaLK-SVM, FaLK-SVMc and FaLK-SVMl vs. LibSVM and FkNN on Large Data Sets

In this experiment we apply FaLK-SVM, FaLK-SVMc, FaLK-SVMl, LibSVM on 8 large data sets comparing the computational and generalisation performances using the RBF kernel, because preliminary experiments showed that the linear or polynomial kernels have very low accuracy results on the considered problems. We also add to the comparison the kNN classifier (implemented with cover trees and called FkNN) using the Euclidean distance.

5.2.1 EXPERIMENTAL PROTOCOL

The data sets considered in this experiment are listed in Table 4 with the corresponding sources and are all scaled in the $[0,1]$ interval. They range from a training set cardinality of about 50k points to more than one million, whereas the dimensionality is not high (always under 60) with separated test sets. In order to select the parameters a 10-fold CV procedure is performed in the training set (apart from FaLK-SVMl) choosing the values in the following sets: $C \in \{2^{-2}, 2^{-1}, \ldots, 2^{9}, 2^{10}\}$, $\sigma \in \{2^{-15}, 2^{-14}, \ldots, 2^{4}, 2^{5}\}$, k for FaLK-SVM in $\{250, 500, 1000, 2000, 4000, 8000\}$ with $k' = k/2$, and k for FkNNSVM in $\{1, 3, 5, 9, 15, 21, 31, 51, 71, 101, 151\}$. FaLK-SVM does not necessarily test all values for k because if the maximum empirical accuracy is found for a specific value of k, for example $k = 500$, and for the following value, in this case $k = 1000$, the maximum is lower, the remaining higher values of k are not tested. Due to the computational resources necessary

data set	FkNN		LibSVM		FaLK-SVM		FaLK-SVMc		FaLK-SVMl
	10f-CV	test	10f-CV	test	10f-CV	test	10f-CV	test	test
ijcnn1	97.37	96.64	98.99	97.98	99.04	**98.04**	98.96	97.98	98.03
cov-type	91.73	91.99	92.60	92.83	92.68	**92.89**	92.44	92.60	92.84
census-inc	94.53	94.52	95.14	**95.13**	95.07	95.07	95.00	94.99	94.99
cod-rna	95.88	96.25	97.18	97.17	97.19	97.23	97.06	97.09	**97.29**
intr-det	99.74	**92.04**	99.89	91.77	99.74	91.97	99.69	92.01	91.91
2-spirals	88.43	88.43	85.18	85.29	88.42	**88.47**	88.29	88.45	88.30
ndcc	85.47	84.99	86.66	86.21	86.63	**86.29**	86.33	85.93	86.24
checker-b	94.31	94.08	94.46	94.21	94.46	94.21	94.45	94.19	**94.23**
test acc. mean rank		4.25		3.25		1.63		3.38	2.50

Table 5: Empirical (using 10-fold CV) and generalisation accuracies of FkNN, LibSVM, FaLK-SVM, FaLK-SVMc and FaLK-SVMl on the 8 large data sets of Experiment 2. The best generalisation accuracy for each data set is highlighted in bold. The last line reports the mean rank of each method among the 8 data sets.

for performing model selection, especially for LibSVM, we performed the cross-validation runs on a Linux-based TORQUE cluster with 20 nodes. For FaLK-SVMl the local model selection is performed on 10 local models, $C \in \{2^0, 2^2, 2^4, 2^6\}$, $k \in \{500, 1000, 2000, 4000\}$, σ locally estimated with the $1st$, $10th$, $50th$ or $90th$ percentile of the distribution of the distances.

5.2.2 RESULTS AND DISCUSSION

Table 5 reports the generalisation accuracies of the analysed classifiers. Looking at the mean ranks, we can see that FaLK-SVM is the most accurate (it achieves the best results in half of the data sets), followed by FaLK-SVMl. LibSVM and FaLK-SVMc seem to perform very similar but little worse than FaLK-SVM and FaLK-SVMl. Not surprisingly, FkNN performs poorly in almost all the data sets, except for the intr-det data set in which it achieves the best result. According to the Wilcoxon Signed Rank Test (Wilcoxon, 1945; Demšar, 2006) FaLK-SVM is significantly more accurate than LibSVM, whereas, excluding FkNN, no other significant differences are detected. Apart for the intr-det data set that has slightly different distribution in the training and testing sets (some types of network attacks are present in the test set only), the best empirical accuracies are always very similar to the generalisation accuracies meaning that all techniques avoid over-fitting.

Table 6 reports the training times together with the speed-ups of FaLK-SVM, FaLK-SVMc and FaLK-SVMl with respect to LibSVM. We can notice that the speed-ups achieved by FaLK-SVM and FaLK-SVMc are always greater than 4.7, and in the majority of the cases they are at least one order of magnitude bigger than LibSVM. Generally, FaLK-SVMc turns out to be faster than FaLK-SVM although the two classifiers implement the same training algorithm. This happens because the model selection chooses for FaLK-SVMc a lower value of k with respect to FaLK-SVM. In fact, FaLK-SVMc is less accurate than FaLK-SVM in choosing the nearest model for a testing point, and this causes an higher value of the $E[\lambda]$ constant that increases the risk of FaLK-SVMc with respect to FaLK-SVM (see Equation 14 and Equation 15). So using a lower k (and thus a lower k') tends to have more

data set	LibSVM	FaLK-SVM		FaLK-SVMc		FaLK-SVMl	
	training time (s)	training time (s)	speed-up on LibSVM	training time (s)	speed-up on LibSVM	train. time with l.m.s.	speed-up on LibSVM
ijcnn1	102	**15**	6.8	**15**	6.8	1850	0.1
cov-type	8362	88	95.0	**38**	220.1	1214	6.9
census-inc	13541	6047	4.7	**2391**	5.7	10271	1.3
cod-rna	9777	395	24.8	**225**	43.5	579	16.9
intr-det	5262	286	18.4	**284**	18.5	450	11.7
2-spirals	4043	188	21.5	**81**	49.9	3442	1.2
ndcc	1487	302	4.9	**92**	16.2	4609	0.3
checker-b	6047	**334**	18.1	366	16.5	1374	4.4

Table 6: Training times for Experiment 2 of LibSVM, FaLK-SVM, FaLK-SVMc and FaLK-SVMl and the speed-ups of the three local methods with respect to LibSVM. The best training time for each data set is highlighted in bold.

data set	LibSVM	FaLK-SVM		FaLK-SVMc		FaLK-SVMl	
	testing time (s)	testing time (s)	speed-up on LibSVM	testing time (s)	speed-up on LibSVM	testing time (s)	speed-up on LibSVM
ijcnn1	43	32	1.3	**5**	8.6	36	1.2
cov-type	2795	202	13.8	**73**	38.3	191	14.6
census-inc	597	1347	0.4	**58**	10.3	1328	0.4
cod-rna	396	261	1.5	**58**	6.8	259	1.5
intr-det	192	146	1.3	**76**	2.5	149	1.3
2-spirals	957	10	95.7	**5**	191.4	18	53.2
ndcc	148	61	2.4	**7**	21.1	61	2.4
checker-b	167	10	16.7	**7**	23.9	**7**	23.9

Table 7: Testing times for Experiment 2 of LibSVM, FaLK-SVM, FaLK-SVMc and FaLK-SVMl and the speed-ups of the three local methods with respect to LibSVM. The best testing time for each data set is highlighted in bold.

models in the proximity of the testing point making the choice less problematic. FaLK-SVMl is sometimes slower than LibSVM, but we have to consider that FaLK-SVMl includes model selection, whereas for the other methods the time needed by model selection is not considered in the training time, so, practically speaking, FaLK-SVMl is the fastest method if the optimal parameters are not *a priori* known.

The testing times required by the analysed methods are reported in Table 7. As expected FaLK-SVMc is the fastest among all methods with speed-up over LibSVM ranging from more than 2 to almost 200. FaLK-SVM and FaLK-SVMl are also generally faster than LibSVM with only one case in which the testing time is about two times slower.

This experiment shows that for 8 non high-dimensional data sets, our approach outperforms a state-of-the-art accurate SVM solver both in terms of generalisation accuracies and computational

performances. Although we have an additional parameter to tune (k), FaLK-SVM and FaLK-SVMc are faster enough to maintain the performance advantages over LibSVM also for model selection (we choose k in a small set of values). Moreover, with FaLK-SVMl we addressed the problem of model selection with a specific approach to set the parameters; FaLK-SVMl outperforms LibSVM in generalisation accuracy, and the time it needs for both internal model selection and training is at least comparable (faster in 7 cases on a total of 8) with the time LibSVM needs for the training only.

5.3 Experiment 3: Comparison of Scalability Performances of FaLK-SVM, FaLK-SVMc, FaLK-SVMl, LibSVM and Approximated SVM Solvers

In this experiment we test the scalability performances of our techniques (FaLK-SVM, FaLK-SVMc, FaLK-SVMl) on training sets with increasing sizes using the RBF kernel against several other techniques. The techniques taken into account are LibSVM, the approximated SVM solvers called CVM, LASVM, USVM, BVM, CPSP (Section 2.3), SVM-bagging with fixed dimension of the sub-sampled training sets (SVM-B) and SVM-bagging with fixed proportion of the sub-sampled training sets with respect to the whole training set (SVM-Bs). Although we apply all the classifiers with the same protocol on the same data sets, we report, for clearness, the results in two parts: the comparison of FaLK-SVM with LibSVM and the approximated SVM solvers in Section 5.3.2, the comparison of FaLK-SVM with its variants FaLK-SVMc and FaLK-SVMl in Section 5.3.3.

5.3.1 EXPERIMENTAL PROTOCOL

We consider here the data sets of Table 4 for which we can further enlarge the training set size. The data sets for which we can add sets of new training examples are the cov-type data set (full training set of 500k points) and the three artificial data sets named 2-spirals, ndcc and checker-b (up to 3 million points). For cov-type the testing set is reduced to 50k examples (the other examples are added to the training set) so the accuracy results are not directly comparable to the previous experiment.

The model selection for all the classifiers (with the exception of FaLK-SVMl that performs internally a local model selection) is performed on the smallest training set only, using the chosen parameter for all the higher training set sizes. This is necessary, especially for LibSVM and approximated SVM solvers, for computational reasons. For LibSVM, BVM, CVM, USVM (with the convex concave procedure) and CPSP, we performed cross validation for C and σ using the same setting of the previous experiment. The default threshold value ε for the stopping criteria are maintained: 10^{-3} for LibSVM, FaLK-SVM, LASVM and 10^{-1} for CPSP while CVM and BVM automatically choose the value of ε based on the data at each application. We set the same size of the kernel cache (100M) for all the methods. The maximum number of core vectors for CVM and BVM is 50000 (the default value), the maximum number of basis vectors for CPSP is set to 1000. For SVM-B and SVM-Bs we need to set respectively the size and the proportion of the sub-sampled training sets and the parameters of the SVMs. For SVM-B the size of the sub-sampled training sets is equal to the 5% of the original training sets of each data set (namely 100000 for cov-type, 2-spirals and ndcc and 300000 for checker-b), whereas SVM-Bs maintains the same sampling rate (5%) for all the applications and so the cardinality of the sub-sampled training sets increases with the cardinality of the training set. Both SVM-B and SVM-Bs train 101 SVMs and the prediction is performed using the majority voting. The value of 101 is chosen because it is a sufficient high number for allowing good accuracies and it is an odd number preventing possible ties in the majority voting. The parameters of the

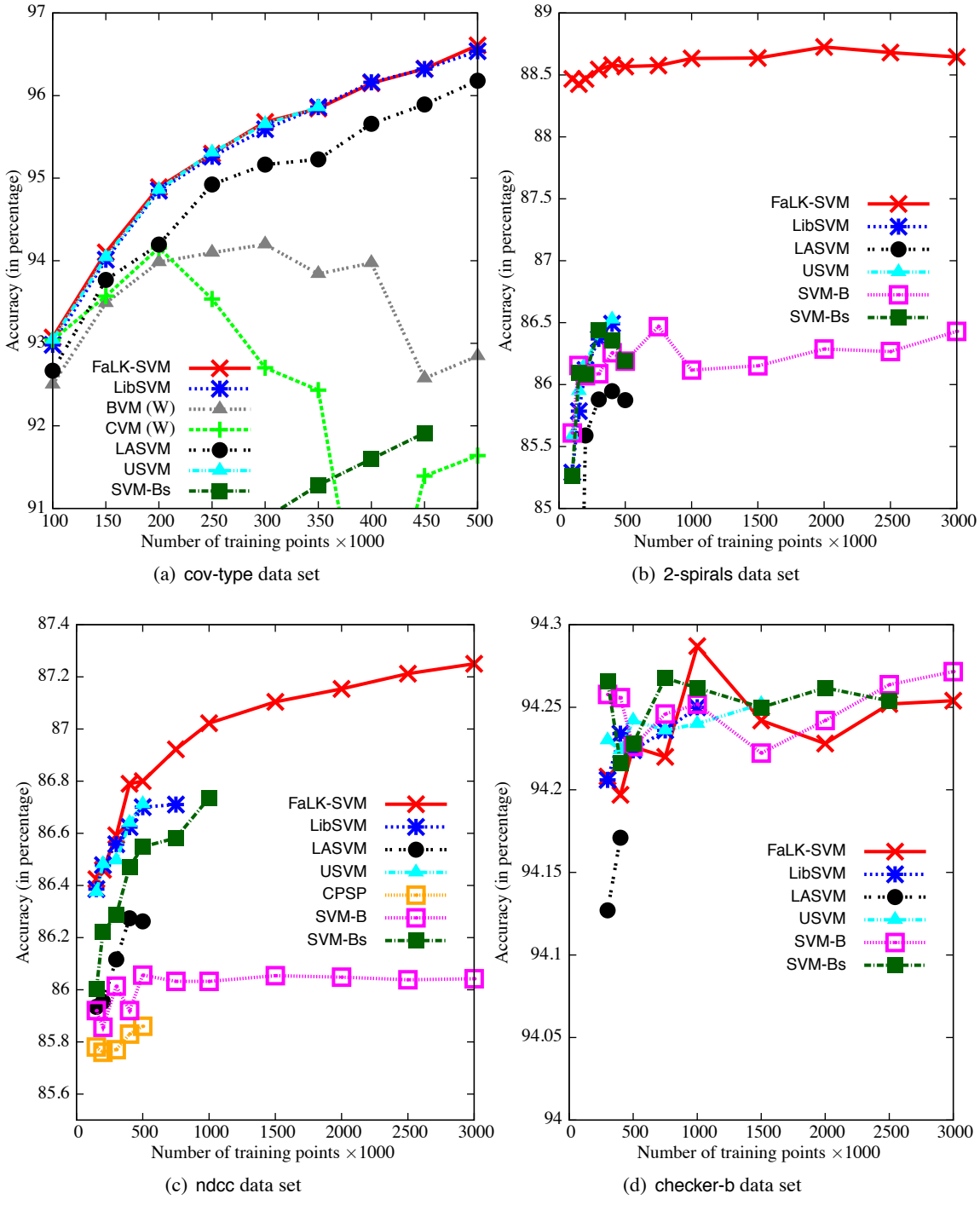
SVM for SVM-B and SVM-Bs (using LibSVM) are chosen using model selection with the same grids of parameters used for LibSVM. We also tested FaLK-SVMl using the same setting of the previous experiment. Each algorithm is tested for training set sizes requiring no more than 100000 seconds (more than 27 hours) for training.

Since the authors of BVM (Tsang et al., 2005) and CVM (Tsang et al., 2007) declared the Linux implementation of their techniques deprecated (see the authors reply to Loosli and Canu (2007) available on BVM webpage), we use the Windows executables on a Intel Pentium D Dual Core CPU 3.40GHz with 2Gb of RAM running Windows XP instead of the AMD Athlon 64 X2 Dual Core Processor 5000+, 2600MHz, with 3.56Gb of RAM with Linux operating system used for all the other classifiers. Because of the use of different operating systems and hardware for BVM and CVM, their running times should not be directly compared to the others. However, the comparison is justified by preliminary tests that showed that the Linux version of BVM on the AMD Athlon machine and the Windows version of BVM on the Intel Pentium machine have similar running times.

5.3.2 RESULTS AND DISCUSSION: FALK-SVM VS LIBSVM AND APPROXIMATED SVM SOLVERS

Figure 3 shows the generalisation accuracies of the methods at increasing training set sizes. Some methods do not appear in the figures due to low generalisation results or computational difficulties that cause abnormal terminations of the algorithms, and some accuracy results for large training set sizes are not present due to the excessive computational time required for training (more than 100000 seconds). We can observe that it is very important to use as many points as possible in order to increase the accuracies for the cov-type and ndcc data sets. The same consideration can be done for the 2-spirals data, although FaLK-SVM already starts from very high accuracies and the increment is limited, while for the checker-b data set the increment of the accuracies is negligible for almost all the methods. For the checker-b data set, the enlarging of the training set is not motivated from the accuracy viewpoint, but we still use it as a benchmark for the computational performances.

Comparing the generalisation accuracies of Figure 3 among the tested methods, we can see that FaLK-SVM is almost always on top for each of the four data sets. In this experiment as well as in the previous ones, we set $k' = k/2$; lower values for k' would probably allow FaLK-SVM to achieve higher accuracy results (although with worse computational performances). However, even if the choice of the k' parameter can be non-optimal, we decided to avoid the model selection for k' since the results are already satisfactory. The methods that seem to give results comparable with FaLK-SVM (apart from the 2-spirals data set) are LibSVM and USVM and they are able, in few cases, to slightly improve the FaLK-SVM results (LibSVM for 2 training set sizes for cov-type and checker-b, USVM for 2 training set sizes for cov-type and checker-b and 1 for ndcc). The bagging techniques give high accuracies only for the checker-b data set; this is not surprising because we already noticed that for the checker-b problem the use of large data sets is not required and thus subsampling-based methods like SVM-B and SVM-Bs are competitive. The results of the online and active learning approach of LASVM are slightly lower than FaLK-SVM, LibSVM and USVM. CPSP gives acceptable results in only one case, and for the 2-spirals and checker-b data sets it suffers from numerical problems possibly due to the scaling of the features in the $[0,1]$ interval. Enlarging the maximum number of basis functions for CPSP gives higher accuracies but the computational time needed to build the models is too high. The results we achieve here for LibSVM and LASVM on the



(a) cov-type data set

(b) 2-spirals data set

(c) ndcc data set

(d) checker-b data set

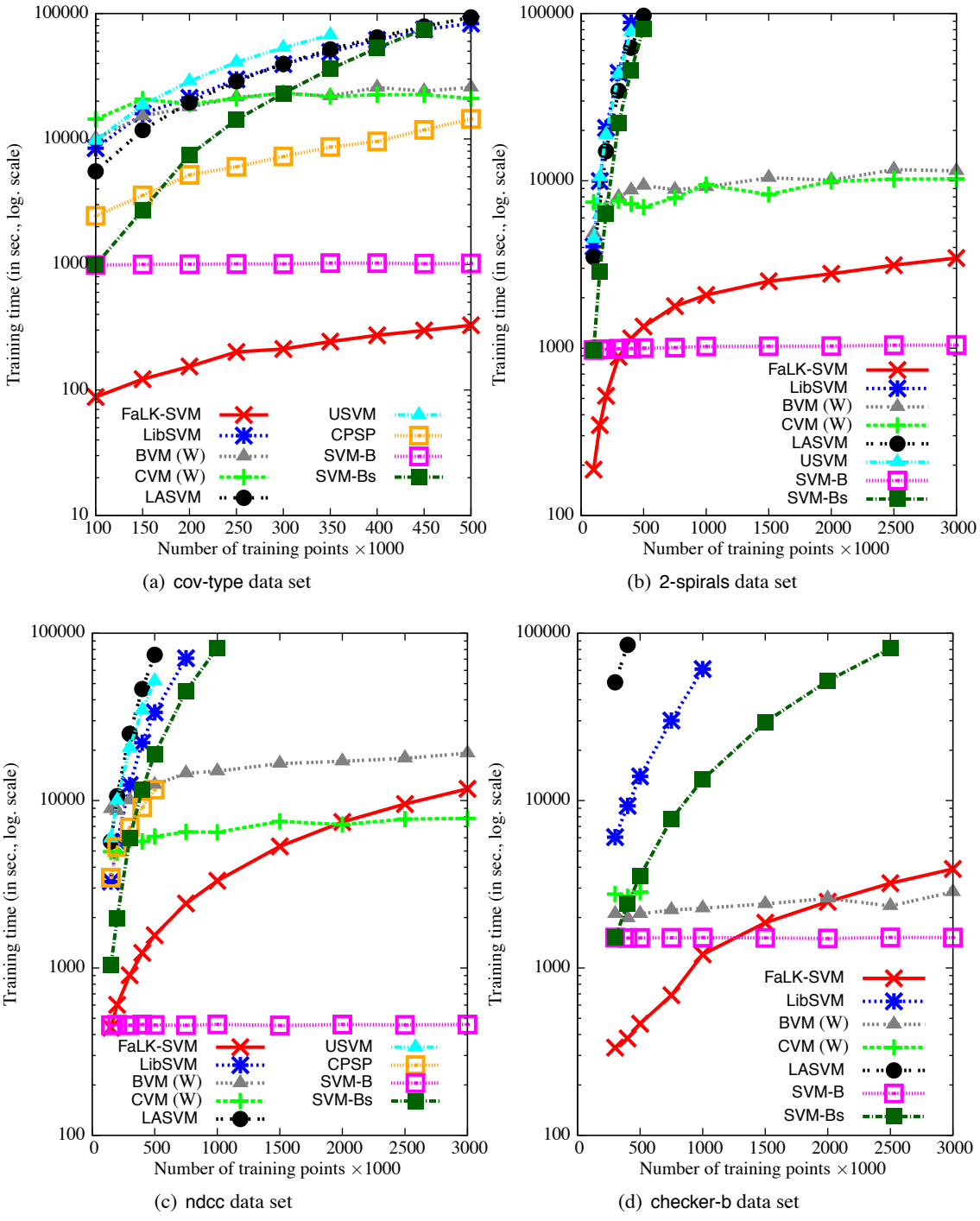
Figure 3: Generalisation accuracies of FaLK-SVM, LibSVM, BVM, CVM, LASVM, USVM, CPSP, SVM-B and SVM-Bs on the cov-type, 2-spirals, ndcc and checker-b data sets with increasing training set sizes (Experiment 3). Some accuracies are missing due to the excessive computational requirements (more than 100000 seconds for training) of the corresponding method for large training set sizes.

cov-type data sets are a little higher than the results in Bordes et al. (2005) (about 1% better both for 100k and 500k training set sizes), and we believe that this is due to the model selection approach we used here that is performed with an exhaustive cross-validation grid search for C and σ. As we can notice in Figure 3, we observed stability problems for CVM and BVM, even if we used the Windows binaries as suggested by the authors.

The training computational performances shown in Figure 4 highlight that FaLK-SVM is always much faster than the alternative techniques that are competitive from the accuracy viewpoint. In fact, although CVM, BVM and SVM-B show good scalability performances and in few cases they overcome the performances of FaLK-SVM, we noticed from Figure 3 that their generalisation abilities are poor. The scaling behaviours of LibSVM, LASVM and USVM are very similar (among the three methods LibSVM is the fastest for ndcc, LASVM is the fastest for 2-spirals and USVM is the fastest for checker-b) but substantially worse than FaLK-SVM one (FaLK-SVM is always at least one order of magnitude faster with speed-ups increasing with the training set sizes). SVM-Bs is slightly faster than LibSVM but the scalability behaviour is very similar. The methods that achieve acceptable accuracy results on the smallest training set size (i.e., LibSVM, LASVM, USVM) are not applicable when the number of training examples increases sensibly because of poor computational scalability performance; this is evident for the 2-spirals, ndcc and checker-b data sets in which the training times of LibSVM, LASVM, USVM exceed 100000 seconds as soon as the training set cardinality approaches one million (the only exception is USVM that is applicable on 1.5 training examples of the checker-b data set). On the contrary, FaLK-SVM processes data sets of 3 millions examples in the order of minutes or few hours. An experiment comparing LibSVM and LASVM on the cov-type data set with conclusions similar to ours is reported by Bordes et al. (2005) in which however LASVM is about a third faster than LibSVM whereas here LibSVM slightly overcomes LASVM; this is probably due to the fact that for LASVM the only available implementation is the original one by Bordes et al. (2005) whereas LibSVM is frequently updated and improved. Finally, CPSP performs slightly better than LibSVM, LASVM and USVM.

The computational performances of the prediction phase are reported in Figure 5. Also in this case the performance of FaLK-SVM is excellent: only CPSP and CVM are faster in 2 data sets than FaLK-SVM, but their corresponding generalisation accuracies are low. As expected, CPSP achieves very fast predictions because it limits the number of basis function to 1000 and thus for each testing points no more than 1000 kernel functions are computed. LibSVM, LASVM and USVM achieve similar results also in testing performances and, apart from small training sets for the ndcc data set, they are at least one order of magnitude slower than FaLK-SVM and the difference grows for large training set sizes. The slowest approach is SVM-Bs and this is due to the high number of models that need to be evaluated for each testing point and to the fact that the size of the models increases with the training set size. Also SVM-B has rather high prediction times but, since the size of the models is almost constant, also the performances at increasing training sizes are constant. It can be argued that the number of models used for bagging can be lowered to obtain faster prediction times; however, if we want to achieve the computational performances of FaLK-SVM we need to use no more than 20 models (in the worst case) and this seriously affects the prediction accuracies that are already much lower than FaLK-SVM ones.

The overall conclusion we can draw about the scalability of the proposed techniques is that, at least for these 4 non high-dimensional data sets, FaLK-SVM is substantially better than the state-of-the-art kernel methods for classification, and this is achieved without affecting the accuracy performances that showed to be always at least as good as the best alternative technique. Apart



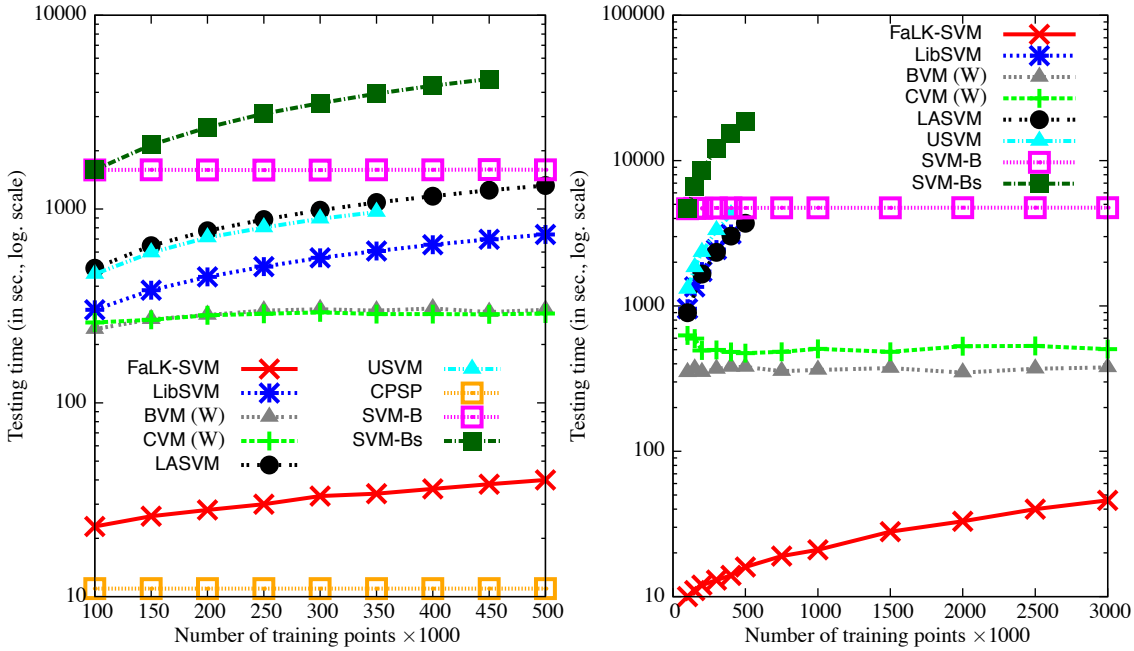
(a) cov-type data set

(b) 2-spirals data set

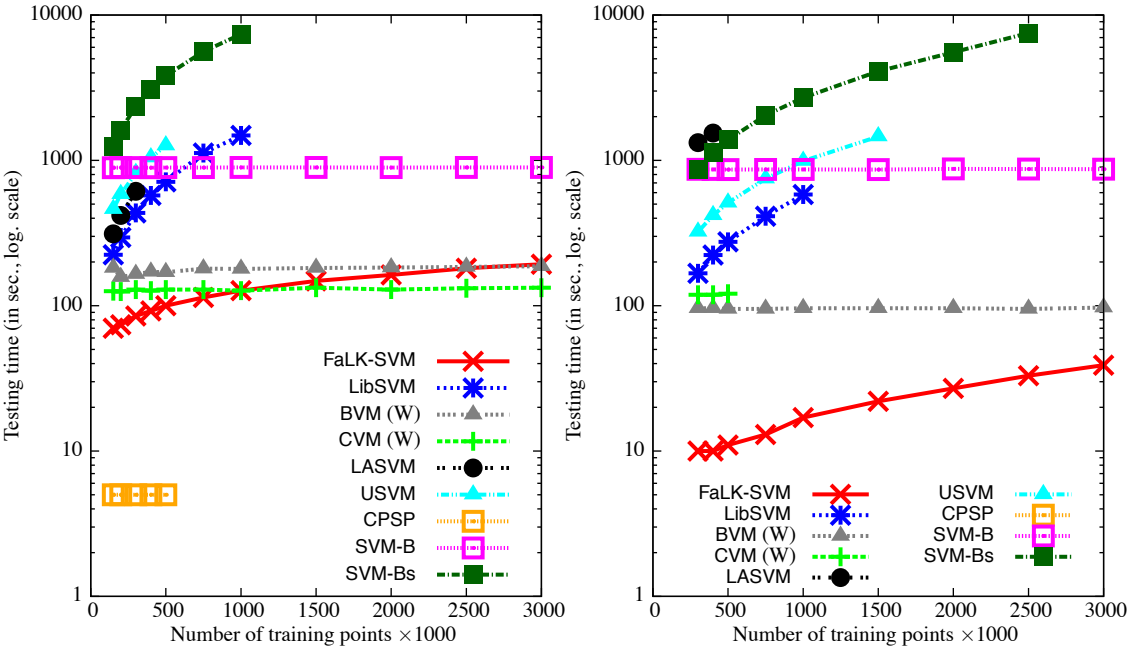
(c) ndcc data set

(d) checker-b data set

Figure 4: Training times of FaLK-SVM, LibSVM, BVM, CVM, LASVM, USVM, CPSP, SVM-B and SVM-Bs on the cov-type, 2-spirals, ndcc and checker-b data sets with increasing training set sizes (Experiment 3). The times (in seconds) are reported in logarithmic scale.



(a) cov-type data set (b) 2-spirals data set



(c) ndcc data set (d) checker-b data set

Figure 5: Testing times of FaLK-SVM, LibSVM, BVM, CVM, LASVM, USVM, CPSP, SVM-B and SVM-Bs on the data sets of Experiment 3 with increasing training set sizes. The times (in seconds) are reported in logarithmic scale. Some testing times are missing due to the excessive computational requirements (more than 100000 seconds for training) of the corresponding method for large training set sizes.

for LibSVM (and consequently for SVM-B and SVM-Bs), we have to say that the available code of the other tested techniques has not been recently updated and for this reason it is possible to argue that higher performances with more optimised implementations of the tested approaches could be reached. It is also necessary to underline that in literature LASVM, USVM, CPSP, BVM and CVM have been prevalently tested on data sets with high dimensionality or, apart for cov-type, on data sets not requiring highly non-linear decision functions. The approximated non-linear SVM solvers and bagging approaches we tested could be indicated for data in which the linear kernel is not the optimal choice, but, at the same time, the decision function can be accurately reconstructed with a reduced amount of information (number of examples, support vectors or basis functions).

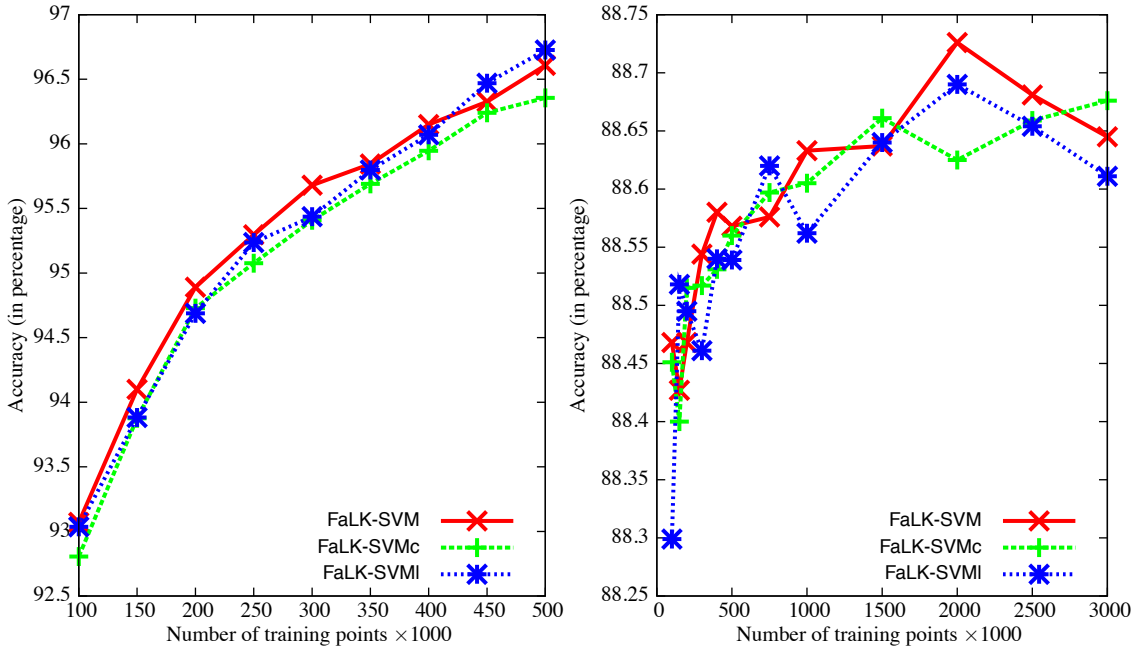
5.3.3 RESULTS AND DISCUSSION: COMPARISON BETWEEN FALK-SVM, FALK-SVMC AND FALK-SVML

Figure 6 reports the comparison of the generalisation accuracies of FaLK-SVM, FaLK-SVMc and FaLK-SVMl at increasing training set size. The computational performances for the training phase are reported in Figure 7, and for the testing phase in Figure 8.

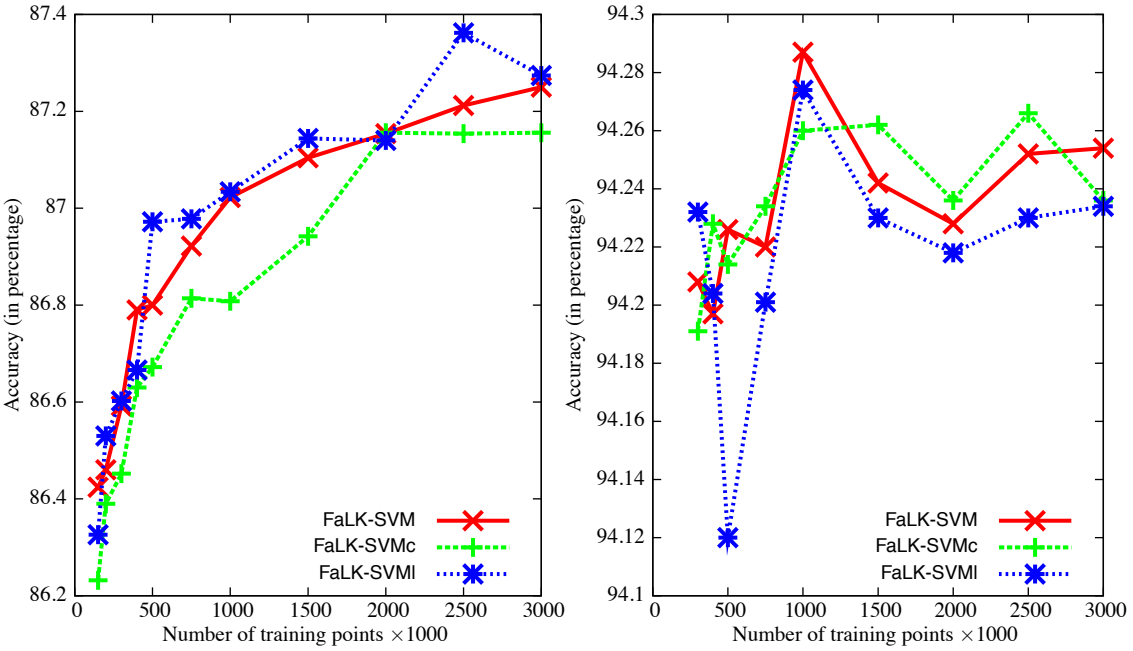
From the accuracy viewpoint, we can notice that, as expected, FaLK-SVM is almost always slightly more accurate than FaLK-SVMc. FaLK-SVMl, apart from checker-b, is less accurate than FaLK-SVM for the smaller training set sizes, and this is due to the fact that FaLK-SVM performs a full grid search for model selection whereas FaLK-SVMl adopts the very fast local model selection approach. However, FaLK-SVMl rivals FaLK-SVM as the training set sizes increases. This is reasonable because FaLK-SVM uses for all the training set sizes the parameters found for the smaller training sets, and the best cross-validated parameters can differ for sub-sampled sets with different cardinality. For example, as the number of training points increases, the radius of the local neighbourhoods decreases if we maintain the same k and k' values, and the original value for the width parameter of the RBF kernel can no longer be the optimal one. For this reason, in the case of cov-type and ndcc data sets, FaLK-SVMl achieves higher accuracies than FaLK-SVM for the largest training sets. FaLK-SVMl shows a slightly higher accuracy variability then FaLK-SVM and FaLK-SVMc at different training set sizes; this is an empirical confirmation that the parameters selected by FaLK-SVMl can be less stable than the parameters selected with standard cross validation, but the phenomenon seems to be acceptable if not negligible.

The training computational performances of Figure 7 confirm (as already discussed in Section 5.2.2) that, although FaLK-SVM and FaLK-SVMc make use of the same training algorithm, the model selection procedure selects lower values of k for FaLK-SVMc, thus assuring faster training times than FaLK-SVM. The speed-ups of FaLK-SVMc with respect to FaLK-SVM are however never higher than one order of magnitude. For FaLK-SVMl we can notice a somehow irregular behaviour for increasing dimensions of the training set and this is due to the different values of the neighbourhood, kernel and regularisation parameters it chooses during the internal fast local model selection phase. In some cases FaLK-SVMl is significantly slower than FaLK-SVM. However, the training times for FaLK-SVMl include the model selection procedure whereas for FaLK-SVM we consider only the training with the optimal parameters, so we can conclude that FaLK-SVMl is a good choice for huge training sets on which traditional model selection becomes intractable.

The testing times reported in Figure 8 confirm that FaLK-SVMc is always faster than FaLK-SVM and FaLK-SVMl. In particular, we can notice that FaLK-SVMc at least halves the testing time of FaLK-SVM. FaLK-SVMl is computationally very similar to FaLK-SVM. This is not surprising because the

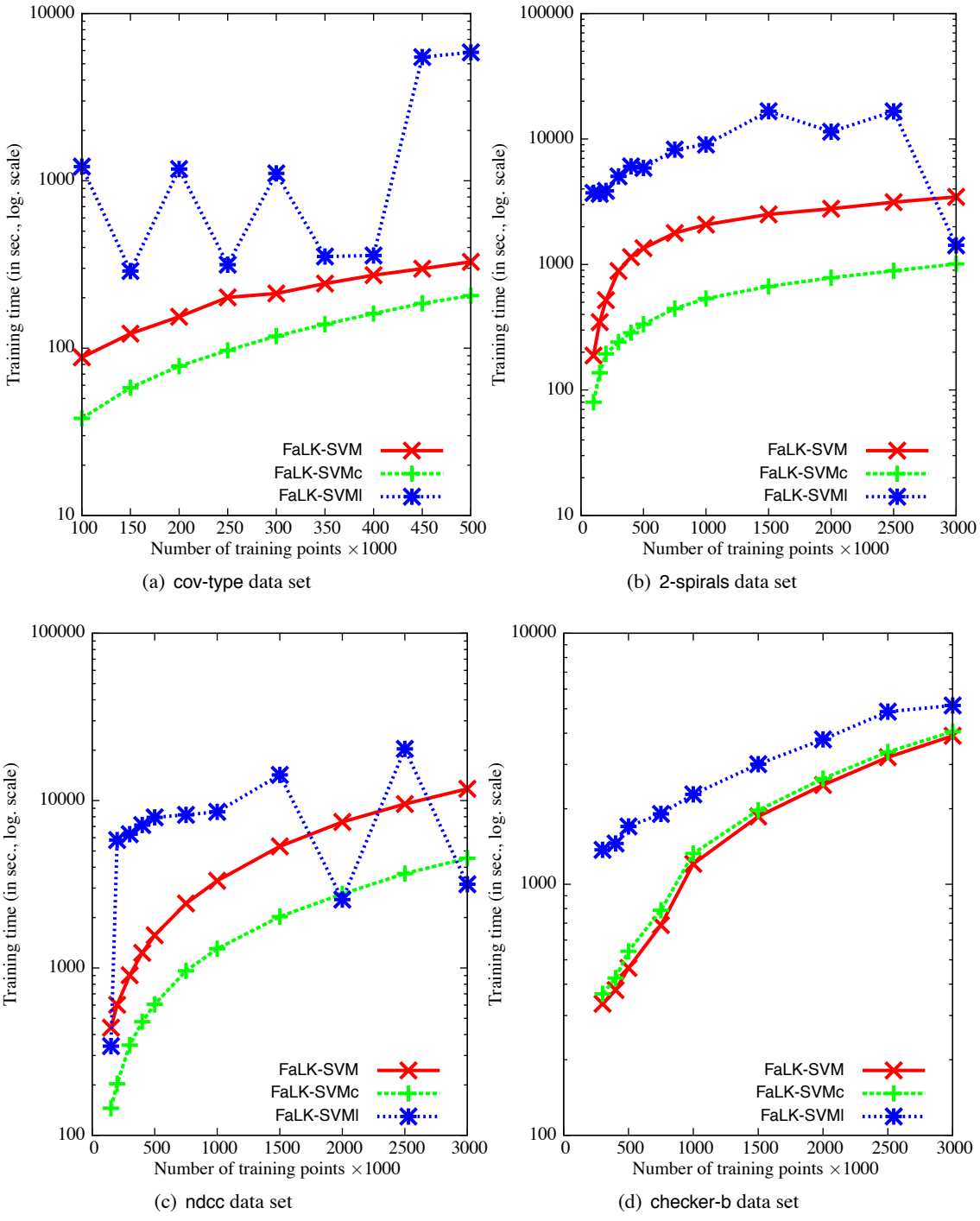


(a) cov-type data set (b) 2-spirals data set



(c) ndcc data set (d) checker-b data set

Figure 6: Generalisation accuracies of FaLK-SVM, FaLK-SVMc and FaLK-SVMl on the cov-type, 2-spirals, ndcc and checker-b data sets with increasing training set sizes (Experiment 3).



(a) cov-type data set

(b) 2-spirals data set

(c) ndcc data set

(d) checker-b data set

Figure 7: Training times of FaLK-SVM, FaLK-SVMc, and FaLK-SVMl on the cov-type, 2-spirals, ndcc and checker-b data sets with increasing training set sizes (Experiment 3). The times (in seconds) are reported in logarithmic scale.

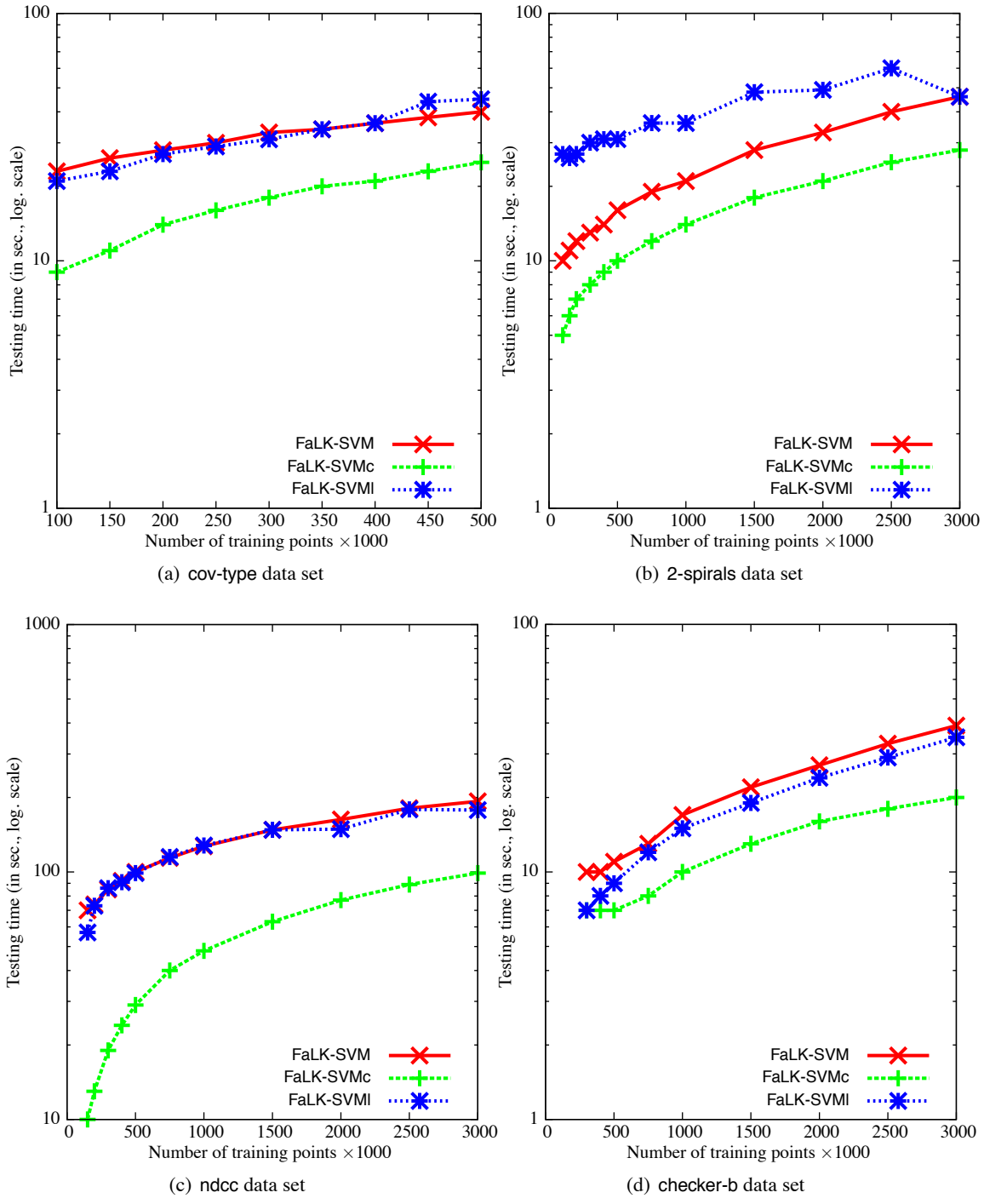


Figure 8: Testing times of FaLK-SVM, FaLK-SVMc, and FaLK-SVMl on the cov-type, 2-spirals, ndcc and checker-b data sets with increasing training set sizes (Experiment 3). The times (in seconds) are reported in logarithmic scale.

only difference between FaLK-SVM and FaLK-SVMl regards the model selection but both classifiers need, during testing, to perform a nearest neighbour search of the query points among all training examples, differently from FaLK-SVMc that performs the nearest neighbour search only among the centres of the local models.

The FaLK-SVMl results permit us to discuss the sensitivity of the local kernel machine approaches with respect to the neighbourhood parameter k. The local model selection of FaLK-SVMl selects different values of k at each application and the k selected by FaLK-SVMl is very often different from the value selected by FaLK-SVM with standard model selection. However, the accuracy results are not very dependent on this as we notice in Figure 6 in which the accuracy variations of FaLK-SVMl as the training sets increase are rather smooth and the accuracies are very similar to FaLK-SVM. So the sensitivity of the accuracies with respect to k for local kernel machines seems to be low at least for large data sets. From the computational viewpoint, instead, we know from the complexity analysis that the advantages of local kernel machines are effective as long as k is substantially lower then N. In our experiments the value of k selected by model selection is bounded to 8000; higher values, although not tested, may decrease the computational performances.

We can conclude that FaLK-SVM, FaLK-SVMc and FaLK-SVMl achieve similar accuracy and computational results. When the model selection for FaLK-SVM and FaLK-SVMc become computationally intractable, FaLK-SVMl is an option to efficiently perform model selection and thus obtain a lower overall training time. When very low testing times are required, FaLK-SVMc is preferable to FaLK-SVM at the price of a slightly lower generalisation accuracy.

6. Conclusions

In this work, we have introduced a new local kernel-based classifier, called FaLK-SVM, that is scalable for large non high-dimensional data. The approach is developed starting from the theory of local learning algorithms and in particular from the Local SVM classifier, called kNNSVM. Various strategies are introduced to overcome the computational problems of kNNSVM and to switch from a completely lazy-learning setting to a eager learning setting with efficient predictions. Learning and complexity bounds for FaLK-SVM are favourable if compared with the SVM ones. FaLK-SVM has, in fact, a training time complexity which is sub-quadratic in the training set size, and a prediction time complexity which is logarithmic. A novel approach for model selection, again based on locality, is introduced obtaining the FaLK-SVMl classifier which substantially unburdens the model selection strategies based on cross-validation. Another variant of the algorithm, calledFaLK-SVMc, permits to simplify the prediction phase. We thus showed that locality can be used to develop computationally efficient classifiers.

We carried out an extensive empirical evaluation of the introduced approaches showing that, for large classification problems requiring non linear decision functions our FaLK-SVM algorithm is much faster and accurate than traditional and approximated SVM solvers. In fact, (i) FaLK-SVM achieves very good accuracy results because it considers all the points without locally under-fitting the data and (ii) FaLK-SVM is very fast and scalable because the cardinality of the local problems can be maintained low. The variant called FaLK-SVMc further enhances testing speed at the price of a little accuracy loss, and the other variant, called FaLK-SVMl, decreases the overall training time.

In general, we have showed that locality can be the key not only for obtaining accurate classifiers, but also for effectively speeding-up kernel-based algorithms.

Further developments of the approach include a dimensionality reduction preprocessing step in order to attack also high-dimensional problems, the application of local classifiers different from SVM, and a distributed parallel version. Also the determination of the critical value of the intrinsic dimensionality (rather than the number of features) above which the local approaches are not effective is still an open question and the answer should be very data-dependent.

References

David W. Aha. *Lazy Learning*. Kluwer Academic Publishers Norwell, MA, USA, 1997.

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary : A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

Arthur Asuncion and David Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The curse of highly variable functions for local kernel machines. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Twenty-third International Conference on Machine Learning (ICML 06)*, pages 97–104, New York, NY, USA, 2006. ACM.

Jock A Blackard and Denis J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24:131–151, 1999.

Enrico Blanzieri and Anton Bryl. Evaluation of the highest probability SVM nearest neighbor classifier with variable relative error cost. In *CEAS 2007*, Mountain View, California, 2007.

Enrico Blanzieri and Farid Melgani. An adaptive SVM nearest neighbor classifier for remotely sensed imagery. In *IEEE International Conference on Geoscience and Remote Sensing Symposium (IGARSS 06)*, pages 3931–3934, 2006.

Enrico Blanzieri and Farid Melgani. Nearest neighbor classification of remote sensing images with the maximal margin principle. *IEEE Transactions on Geoscience and Remote Sensing*, 46(6): 1804–1811, 2008.

Antoine Bordes and Léon Bottou. The Huller: A simple and efficient online SVM. In *Machine Learning: ECML 2005*, Lecture Notes in Artificial Intelligence, LNAI 3720, pages 505–512. Springer Verlag, 2005.

Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

Antoine Bordes, Léon Bottou, and Patrick Gallinari. SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, July 2009.

Léon Bottou and Vladimir N. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

Léon Bottou, Corinna Cortes, John S. Denker, Harris Drucker, Isabelle Guyon, Lawrence D. Jackel, Yann Le Cun, Urs A. Muller, Eduard Säckinger, Patrice Simard, and Vladimir Vapnik. Comparison of classifier methods: A case study in handwritten digit recognition. In *Twelveth IAPR International Conference on Pattern Recognition, Conference B: Computer Vision & Image Processing*, volume 2, pages 77–82. IEEE, 1994.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

David Broomhead and David Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale l2-loss linear support vector machines. *Journal of Machine Learning Research*, 9:1369–1398, 2008.

Qun Chang, Qingcai Chen, and Xiaolong Wang. Scaling Gaussian RBF kernel width to improve SVM classification. *International Conference on Neural Networks and Brain, 2005. (ICNN&B 05)*, 1:19–22, 2005.

Long Chen. New analysis of the sphere covering problems and optimal polytope approximation of convex bodies. *Journal of Approximation Theory*, 133(1):134, 2005.

Haibin Cheng, Pang-Ning Tan, and Rong Jin. Localized support vector machine and its efficient algorithm. *SIAM International Conference on Data Mining*, 2007.

Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, pages 233–235, 1979.

Kenneth L. Clarkson. Nearest neighbor queries in metric spaces. In *Twenty-ninth Annual ACM Symposium on Theory of computing (STOC 97)*, pages 609–617, New York, NY, USA, 1997. ACM.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9:1775–1822, 2008.

Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Trading convexity for scalability. In *Twenty-third International Conference on Machine Learning (ICML 06)*, pages 201–208, New York, NY, USA, 2006. ACM.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Jian-xiong Dong. Fast SVM training algorithm with decomposition on very large data sets. *IEEE Transaction Pattern Analysis and Machine Intelligence*, 27(4):603–618, 2005. Senior Member-Krzyzak, Adam and Fellow-Suen, Ching Y.

Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *The Journal of Machine Learning Research*, 6: 1889–1918, 2005.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman & Co. New York, NY, USA, 1979.

Seth Hettich and Stephen D. Bay. The UCI KDD archive, 1999. URL `http://kdd.ics.uci.edu`.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Twenty-fifth International Conference on Machine Learning (ICML 08)*, pages 408–415, New York, NY, USA, 2008. ACM.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

Thorsten Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

Thorsten Joachims. Training linear SVMs in linear time. In *Twelveth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226. ACM New York, NY, USA, 2006.

Thorsten Joachims and Chun-Nam Yu. Sparse kernel SVMs via cutting-plane training. *Machine Learning*, 2009.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press Cambridge, MA, USA, 1994.

Sathiya Keerthi and Dennis DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493–1515, 2006.

Stefan Knerr, Leon Personnaz, and Gerard Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. *Optimization Methods and Software*, 1: 23–34, 1990.

Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In *Fifteenth Annual ACM-SIAM Symposium on Discrete algorithms (SODA 04)*, pages 798–807, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

Ulrich H.-G. Kressel. Pairwise classification and support vector machines. *Advances in Kernel Methods: Support Vector Learning*, pages 255–268, 1999.

Yuh-jye Lee and Olvi L. Mangasarian. RSVM: Reduced support vector machines. In *First SIAM International Conference on Data Mining*, 2001.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton methods for large-scale logistic regression. In *Twenty-fourth International Conference on Machine learning (ICML 07)*, pages 561–568, New York, NY, USA, 2007. ACM.

Gaëlle Loosli and Stéphane Canu. Comments on the "Core vector machines: Fast SVM training on very large data sets". *Journal of Machine Learning Research*, 8:291–301, 2007.

Olvi L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

Mario Marchand and John Shawe-Taylor. The set covering machine. *The Journal of Machine Learning Research*, 3:723–746, 2003.

Donald Michie, David J. Spiegelhalter, Charles C. Taylor, and John Campbell, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Upper Saddle River, NJ, USA, 1994. ISBN 0-13-106360-X.

Paul Nemenyi. *Distribution-Free Multiple Comparisons*. PhD thesis, Princeton, 1963.

John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems*, 12(3):547–553, 2000.

Bernard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press Cambridge, MA, USA, 2002.

Nicola Segata. FaLKM-lib v1.0: A library for fast local kernel machines. Technical Report DISI-09-025, id 1613, DISI, University of Trento, Italy, 2009. Software available at `http://disi.unitn.it/~segata/FaLKM-lib`.

Nicola Segata and Enrico Blanzieri. Empirical assessment of classification accuracy of Local SVM. In *Eighteenth Annual Belgian-Dutch Conference on Machine Learning (Benelearn 2009)*, pages 47–55, 2009a.

Nicola Segata and Enrico Blanzieri. Operators for transforming kernels into quasi-local kernels that improve SVM accuracy. Technical Report DISI-09-042, id 1652, Tech. rep., DISI, University of Trento, 2009b.

Nicola Segata and Enrico Blanzieri. Fast local support vector machines for large datasets. In *International Conference on Machine Learning and Data Mining (MLDM 2009)*, volume 5632 of *Lecture Notes in Computer Science*. Springer, 2009c.

Nicola Segata, Enrico Blanzieri, and Pádraig Cunningham. A scalable noise reduction technique for large case-based systems. In L Ginty and D.C Wilson, editors, *Case-Based Reasoning Research and Development: 8th International Conference on Case-Based Reasoning (ICCBR09)*, volume 09 of *Lecture Notes in Artificial Intelligence*, pages 755–758. Springer, 2009a.

Nicola Segata, Enrico Blanzieri, Sarah Jane Delany, and Pádraig Cunningham. Noise reduction for instance-based learning with a local maximal margin approach. *Journal of Intelligent Information Systems*, 2009b. In Press.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Twenty-fourth International Conference on Machine learning (ICML 07)*, pages 807–814, New York, NY, USA, 2007. ACM.

Alexander J. Smola, SVN Vishwanathan, and Quoc V. Le. Bundle methods for machine learning. *Advances in Neural Information Processing Systems*, 20:1377–1384, 2008.

Michael E. Thompson. NDCC: Normally distributed clustered datasets on cubes, 2006. www.cs.wisc.edu/dmi/svm/ndcc/.

Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast SVM training on very large data sets. *The Journal of Machine Learning Research*, 6:363–392, 2005.

Ivor W. Tsang, Andras Kocsor, and James T. Kwok. Simpler core vector machines with enclosing balls. In *Twenty-fourth International Conference on Machine Learning (ICML 07)*, pages 911–918, New York, NY, USA, 2007. ACM.

Andrew V. Uzilov, Joshua M. Keegan, and David H. Mathews. Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change. *BMC Bioinformatics*, 7(1): 173, 2006.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

Vladimir N. Vapnik and Léon Bottou. Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 5(6):893–909, 1993.

Jigang Wang, Predrag Neskovic, and N. Leon Cooper. A minimum sphere covering approach to pattern classification. *International Conference on Pattern Recognition*, 3:433–436, 2006.

Xun-Kai Wei and Ying-Hong Li. Linear programming minimum sphere set covering for extreme learning machines. *Neurocomputing*, 71(4–6):570–575, 2008.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

Tao Yang and Vojislav Kecman. Adaptive local hyperplane classification. *Neurocomputing*, 71 (13-15):3001–3004, 2008.

Tao Yang and Vojislav Kecman. Adaptive local hyperplane algorithm for learning small medical data sets. *Expert Systems*, 26(4):355–359, 2009.

Tao Yang and Vojislav Kecman. Face recognition with adaptive local hyperplane algorithm. *Pattern Analysis & Applications*, 13(1):79–83, 2010. ISSN 1433-7541.

Alan L. Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4): 915–936, 2003.

Alon Zakai and Ya'acov Ritov. Consistency and localizability. *Journal of Machine Learning Research*, 10:827–856, 2009.

Luca Zanni, Thomas Serafini, and Gaetano Zanghirati. Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research*, 7: 1467–1492, 2006.

Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:2126–2136, 2006.

Journal of Machine Learning Research 11 (2010) 1927-1956 Submitted 9/09; Revised 5/10; Published 7/10

Chromatic PAC-Bayes Bounds for Non-IID Data: Applications to Ranking and Stationary β-Mixing Processes

Liva Ralaivola LIVA.RALAIVOLA@LIF.UNIV-MRS.FR
Laboratoire d'Informatique Fondamentale de Marseille
CNRS UMR 6166
Aix-Marseille Université
39, rue Joliot Curie
F-13013 Marseille, France

Marie Szafranski MARIE.SZAFRANSKI@IBISC.FR
Informatique, Biologie Intégrative et Systèmes Complexes
CNRS EA 4526
Université d'Evry Val d'Essonne
523 pl. des terrasses de l'Agora
F-91000 Évry, France

Guillaume Stempfel GUILLAUME.STEMPFEL@LIF.UNIV-MRS.FR
Laboratoire d'Informatique Fondamentale de Marseille
CNRS UMR 6166
Aix-Marseille Université
39, rue Joliot Curie
F-13013 Marseille, France

Editor: Nicolas Vayatis

Abstract

PAC-Bayes bounds are among the most accurate generalization bounds for classifiers learned from independently and identically distributed (IID) data, and it is particularly so for margin classifiers: there have been recent contributions showing how practical these bounds can be either to perform model selection (Ambroladze et al., 2007) or even to directly guide the learning of linear classifiers (Germain et al., 2009). However, there are many practical situations where the training data show some dependencies and where the traditional IID assumption does not hold. Stating generalization bounds for such frameworks is therefore of the utmost interest, both from theoretical and practical standpoints. In this work, we propose the first—to the best of our knowledge—PAC-Bayes generalization bounds for classifiers trained on data exhibiting interdependencies. The approach undertaken to establish our results is based on the decomposition of a so-called dependency graph that encodes the dependencies within the data, in sets of independent data, thanks to graph *fractional covers*. Our bounds are very general, since being able to find an upper bound on the fractional chromatic number of the dependency graph is sufficient to get new PAC-Bayes bounds for specific settings. We show how our results can be used to derive bounds for ranking statistics (such as AUC) and classifiers trained on data distributed according to a stationary β-mixing process. In the way, we show how our approach seamlessly allows us to deal with U-processes. As a side note, we also provide a PAC-Bayes generalization bound for classifiers learned on data from stationary φ-mixing distributions.

Keywords: PAC-Bayes bounds, non IID data, ranking, U-statistics, mixing processes

1. Introduction

This introductory section first recalls a few PAC-Bayesian results that set the background of the present work. An overview of our contributions is then given, and results from the literature closely related to ours are described. Lastly, the structure of the paper is provided.

1.1 Background

Recently, there has been much progress in the field of generalization bounds for classifiers, the most noticeable of which are Rademacher-complexity-based bounds (Bartlett and Mendelson, 2002; Bartlett et al., 2005), stability-based bounds (Bousquet and Elisseeff, 2002) and PAC-Bayes bounds (McAllester, 1999). PAC-Bayes bounds, introduced by McAllester (1999), and refined in several occasions (Seeger, 2002a; Langford, 2005; Audibert and Bousquet, 2007), are some of the most appealing advances from the tightness and accuracy points of view (an excellent monograph on the PAC-Bayesian framework is that of Catoni (2007)). Among others, striking results have been obtained concerning PAC-Bayes bounds for linear classifiers: Ambroladze et al. (2007) showed that PAC-Bayes bounds are a viable route to do actual model selection; Germain et al. (2009) recently proposed to learn linear classifiers by directly minimizing the linear PAC-Bayes bound with conclusive results, while Langford and Shawe-taylor (2002) showed that under some margin assumption, the PAC-Bayes framework allows one to tightly bound not only the risk of the stochastic Gibbs classifier (see below) but also the risk of the Bayes classifier. The variety of (algorithmic, theoretical, practical) outcomes that can be expected from original contributions in the PAC-Bayesian setting explains and justifies the increasing interest it generates.

1.2 Contribution

To the best of our knowledge, PAC-Bayes bounds have essentially been derived for the setting where the training data are *independently and identically distributed* (IID). Yet, being able to learn from non-IID data while having strong theoretical guarantees on the generalization properties of the learned classifier is an actual problem in a number of real world applications such as, for instance, bipartite ranking (and more generally k-partite ranking) or classification from sequential data. Here, we propose the first PAC-Bayes bounds for classifiers trained on non-IID data; they constitute a generalization of the IID PAC-Bayes bound and they are generic enough to provide a principled way to establish generalization bounds for a number of non-IID settings. To establish these bounds, we make use of simple tools from probability theory, convexity properties of some functions, and we exploit the notion of *fractional covers* of graphs (Schreinerman and Ullman, 1997). One way to get a high level view of our contribution is the following: fractional covers allow us to cope with the dependencies within the set of random variables at hand by providing a strategy to make (large) subsets of independent random variables on which the usual IID PAC-Bayes bound is applied. Note that we essentially provide bounds for the case of *identically and non-independently* distributed data; the additional results that we give in the appendix generalizes to *non-identically and non-independently* distributed data.

1.3 Related Results

We would like to mention that the idea of dealing with sums of interdependent random variables by separating them into subsets of independent variables to establish concentration inequalities dates

back to the work of Hoeffding (1948, 1963) on U-statistics. Explicitly using the notion of (fractional) covers—or, equivalently, colorings—of graphs to derive such concentration inequalities has been proposed by Pemmaraju (2001) and Janson (2004) and later extended by Usunier et al. (2006) to deal with functions that are different from the sum. Just as Usunier et al. (2006), who used their concentration inequality to provide generalization bounds based on the *fractional Rademacher complexity*, we take the approach of decomposing a set of dependent random variables into subsets of dependent random variables a step beyond establishing concentration inequality to provide what we call *chromatic* PAC-Bayes generalization bounds.

The genericity of our bounds is illustrated in several ways. It allows us to derive generalization bounds on the ranking performance of scoring/ranking functions using two different performance measures, among which the *area under the* ROC *curve* (AUC) . These bounds are directly related to the work of Agarwal et al. (2005), Agarwal and Niyogi (2009), Clémençon et al. (2008) and Freund et al. (2003). Even if our bounds are obtained as simple specific instances of our generic PAC-Bayes bounds, they exhibit interesting peculiarities. Compared with the bound of Agarwal et al. (2005) and Freund et al. (2003), our AUC bound depends in a less stronger way on the *skew* (that is, the imbalance between positive and negative data) of the distribution; besides it does not rest on (rank-)shatter coefficients/VC dimension that may sometimes be hard to assess accurately; in addition, our bound directly applies to (kernel-based) linear classifiers. Agarwal and Niyogi (2009) base their analysis of ranking performances on algorithmic stability, and the qualitative comparison of their bounds and ours is not straightforward because stability arguments are somewhat different from the arguments used for PAC-Bayes bounds (and other uniform bounds). As already observed by Janson (2004), coloring provides a way to generalize large deviation results based on U-statistics; this observation carries over when generalization bounds are considered, which allows us to draw a connection between the results we obtain and that of Clémençon et al. (2008).

Another illustration of the genericity of our approach deals with mixing processes. In particular, we show how our chromatic bounds can be used to easily derive new generalization bounds for β-mixing processes. Rademacher complexity based bounds for such type of processes have recently been established by Mohri and Rostamizadeh (2009). To the best of our knowledge, it is the first time that such a bound is given in the PAC-Bayes framework. The striking feature is that it is done at a very low price: the independent block method proposed by Yu (1994) directly gives a dependency graph whose chromatic number is straightforward to compute. As we shall see, this suffices to instantiate our chromatic bounds, which, after simple calculations, leads to appropriate generalization bound. For sake of completeness, we also provide a PAC-Bayes bound for stationary φ-mixing processes; it is based on a different approach and its presentation is postponed to the appendix together with the tools that allows us to derive it.

1.4 Organization of the Paper

The paper is organized as follows. Section 2 recalls the standard IID PAC-Bayes bound. Section 3 introduces the notion of fractional covers and states the new chromatic PAC-Bayes bounds, which rely on the fractional chromatic number of the *dependency graph* of the data at hand. Section 4 provides specific versions of our bounds for the case of IID data, ranking and stationary β-mixing processes, giving rise to original generalization bounds. A PAC-Bayes bound for stationary φ-mixing based on arguments different from the chromatic PAC-Bayes bound is provided, in the appendix.

2. IID PAC-Bayes Bound

We introduce notation that will hold from here on. We mainly consider the problem of binary classification over the *input space* $\mathcal{X}$ and we denote the set of possible labels as $\mathcal{Y} = \{-1, +1\}$ (for the case of ranking described in Section 4, we we use $\mathcal{Y} = \mathcal{R}$); $\mathcal{Z}$ denotes the product space $\mathcal{X} \times \mathcal{Y}$. $\mathcal{H} \subseteq \mathcal{R}^{\mathcal{X}}$ is a family of real valued classifiers defined on $\mathcal{X}$: for $h \in \mathcal{H}$, the predicted output of $x \in \mathcal{X}$ is given by $\text{sign}(h(x))$, where $\text{sign}(x) = +1$ if $x \geq 0$ and -1 otherwise. D is a probability distribution defined over $\mathcal{Z}$ and $\mathbf{D}_m$ denotes the distribution of an m-sample; for instance, $\mathbf{D}_m = \otimes_{i=1}^m D = D^m$ is the distribution of an IID sample $\mathbf{Z} = \{Z_i\}_{i=1}^m$ of size m ($Z_i \sim D$, $i = 1 \ldots m$). P and Q are distributions over $\mathcal{H}$. For any positive integer m, $[m]$ stands for $\{1, \ldots, m\}$.

The IID PAC-Bayes bound, can be stated as follows (McAllester, 2003; Seeger, 2002a; Langford, 2005).

Theorem 1 (IID PAC-Bayes Bound) *$\forall D$, $\forall \mathcal{H}$, $\forall \delta \in (0,1]$, $\forall P$, with probability at least $1-\delta$ over the random draw of $\mathbf{Z} \sim \mathbf{D}_m = D^m$, the following holds:*

$$\forall Q,\ \text{kl}(\hat{e}_Q(\mathbf{Z}) || e_Q) \leq \frac{1}{m}\left[\text{KL}(Q||P) + \ln \frac{m+1}{\delta}\right]. \tag{1}$$

This theorem provides a generalization error bound for the *Gibbs classifier* g_Q: given a distribution Q, this stochastic classifier predicts a class for $\mathbf{x} \in \mathcal{X}$ by first drawing a hypothesis h according to Q and then outputting $\text{sign}(h(\mathbf{x}))$. Here, $\hat{e}_Q$ is the empirical error of g_Q on an IID sample $\mathbf{Z}$ of size m and e_Q is its true error:

$$\begin{aligned} \hat{e}_Q(\mathbf{Z}) &:= \mathbb{E}_{h\sim Q} \frac{1}{m}\sum_{i=1}^m r(h, Z_i) = \mathbb{E}_{h\sim Q}\hat{R}(h,\mathbf{Z}) \quad && \text{with } \hat{R}(h,\mathbf{Z}) := \tfrac{1}{m}\textstyle\sum_{i=1}^m r(h,Z_i), \\ e_Q &:= \mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m}\hat{e}_Q(\mathbf{Z}) = \mathbb{E}_{h\sim Q} R(h) && \text{with } R(h) := \mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m}\hat{R}(h,\mathbf{Z}), \end{aligned} \tag{2}$$

where, for $Z = (X, Y)$,

$$r(h, Z) := \mathbb{I}_{Yh(X)<0}.$$

Note that we will use this binary 0-1 risk function r throughout the paper and that a generalization of our results to bounded real-valued risk functions is given in appendix. Since $\mathbf{Z}$ is an (independently) identically distributed sample, we have

$$R(h) = \mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m}\hat{R}(h,\mathbf{Z}) = \mathbb{E}_{Z\sim D} r(h,Z). \tag{3}$$

For $p, q \in [0,1]$, $\text{kl}(q||p)$ is the Kullback-Leibler divergence between the Bernoulli distributions with probabilities of success q and p, and $\text{KL}(Q||P)$ is the Kullback-Leibler divergence between Q and P:

$$\begin{aligned} \text{kl}(q||p) &:= q \ln \frac{q}{p} + (1-q)\ln\frac{1-q}{1-p}, \\ \text{KL}(Q||P) &:= \mathbb{E}_{h\sim Q} \ln \frac{Q(h)}{P(h)}, \end{aligned}$$

where $\text{kl}(0||0) = \text{kl}(1||1) = 0$. All along, we assume that the posteriors are absolutely continuous with respect to their corresponding priors.

It is straightforward to see that the mapping $\mathrm{kl}_q : t \mapsto \mathrm{kl}(q||q+t)$ is strictly increasing for $t \in [0, 1-q)$ and therefore defines a bijection from $[0, 1-q)$ to $\mathcal{R}_+$: we denote by kl_q^{-1} its inverse. Then, as pointed out by Seeger (2002a), the function $\mathrm{kl}^{-1} : (q, \varepsilon) \mapsto \mathrm{kl}^{-1}(q, \varepsilon) = \mathrm{kl}_q^{-1}(\varepsilon)$ is well-defined over $[0,1) \times \mathcal{R}^+$, and, by definition:

$$t \geq \mathrm{kl}^{-1}(q, \varepsilon) \Leftrightarrow \mathrm{kl}(q||q+t) \geq \varepsilon.$$

This makes it possible to rewrite bound (1) in a more 'usual' form:

$$\forall Q,\ e_Q \leq \hat{e}_Q(\mathbf{Z}) + \mathrm{kl}^{-1}\left(\hat{e}_Q(\mathbf{Z}), \frac{1}{m}\left[\mathrm{KL}(Q||P) + \ln\frac{m+1}{\delta}\right]\right). \tag{4}$$

We observe that even if bounds (1) and (4) apply to the risk e_Q of the stochastic classifier g_Q, a straightforward argument gives that, if b_Q is the (deterministic) Bayes classifier such that $b_Q(x) = \mathrm{sign}(\mathbb{E}_{h\sim Q} h(x))$, then $R(b_Q) = \mathbb{E}_{Z\sim D} r(b_Q, Z) \leq 2e_Q$ (see for instance Herbrich and Graepel, 2001). Langford and Shawe-taylor (2002) show that under some margin assumption, $R(b_Q)$ can be bounded even more tightly.

3. Chromatic PAC-Bayes Bounds

The problem we focus on is that of generalizing Theorem 1 to the situation where there may exist probabilistic dependencies between the elements Z_i of $\mathbf{Z} = \{Z_i\}_{i=1}^m$ while the marginal distributions of the Z_i's are identical. As announced before, we provide PAC-Bayes bounds for classifiers trained on identically but not independently distributed data. These results rely on properties of a dependency graph that is built according to the dependencies within $\mathbf{Z}$. Before stating our new bounds, we thus introduce the concepts of graph theory that will play a role in their statements.

3.1 Dependency Graph, Fractional Covers

Definition 2 (Dependency Graph) *Let $\mathbf{Z} = \{Z_i\}_{i=1}^m$ be a set of m random variables taking values in some space $\mathcal{Z}$. The* dependency graph $\Gamma(\mathbf{Z}) = (V, E)$ *of $\mathbf{Z}$ is such that:*

- *the set of vertices V of $\Gamma(\mathbf{Z})$ is $V = [m]$;*
- *$(i, j) \notin E$ (there is no edge between i and j) $\Leftrightarrow$ Z_i and Z_j are independent.*

Definition 3 (Fractional Covers, Schreinerman and Ullman, 1997) *Let $\Gamma = (V, E)$ be an undirected graph, with $V = [m]$.*

- *$C \subseteq V$ is* independent *if the vertices in C are independent (no two vertices in C are connected).*
- *$\mathbf{C} = \{C_j\}_{j=1}^n$, with $C_j \subseteq V$, is a* proper cover *of V if each C_j is independent and $\bigcup_{j=1}^n C_j = V$. It is* exact *if $\mathbf{C}$ is a partition of V. The size of $\mathbf{C}$ is n.*
- *$\mathbf{C} = \{(C_j, \omega_j)\}_{j=1}^n$, with $C_j \subseteq V$ and $\omega_j \in [0,1]$, is a* proper exact fractional cover *of V if each C_j is independent and $\forall i \in V$, $\sum_{j=1}^n \omega_j \mathbb{I}_{i\in C_j} = 1$; $\omega(\mathbf{C}) = \sum_{j=1}^n \omega_i$ is the* chromatic weight *of $\mathbf{C}$.*
- *The (fractional) chromatic number $\chi(\Gamma)$ ($\chi^*(\Gamma)$) is the minimum size (chromatic weight) over all proper exact (fractional) covers of Γ*

A cover is a fractional cover such that all the weights ω_i are equal to 1 (and all the results we state for fractional covers apply to the case of covers). If n is the size of a cover, it means that the nodes of the graph at hand can be colored with n colors in a way such that no two adjacent nodes receive the same color.

The problem of computing the (fractional) chromatic number of a graph is NP-hard (Schreinerman and Ullman, 1997). However, for some particular graphs as those that come from the settings we study in Section 4, this number can be evaluated precisely. If it cannot be evaluated, it can be upper bounded using the following property.

Property 1 (Schreinerman and Ullman, 1997) *Let $\Gamma = (V,E)$ be a graph. Let $c(\Gamma)$ be the* clique number *of Γ, that is, the order of the largest clique in Γ. Let $\Delta(\Gamma)$ be the maximum degree of a vertex in Γ. We have the following inequalities:*

$$1 \leq c(\Gamma) \leq \chi^*(\Gamma) \leq \chi(\Gamma) \leq \Delta(\Gamma) + 1.$$

In addition, $1 = c(\Gamma) = \chi^*(\Gamma) = \chi(\Gamma) = \Delta(\Gamma) + 1$ if and only if Γ is totally disconnected.

If $\mathbf{Z} = \{Z_i\}_{i=1}^m$ is a set of random variables over $\mathcal{Z}$ then a (fractional) proper cover of $\Gamma(\mathbf{Z})$, splits $\mathbf{Z}$ into subsets of independent random variables. This is a crucial feature to establish our results. In addition, we can see $\chi^*(\Gamma(\mathbf{Z}))$ and $\chi(\Gamma(\mathbf{Z}))$ as measures of the amount of dependencies within $\mathbf{Z}$.

The following lemma (Lemma 3.1 in Janson, 2004) will be very useful in the following.

Lemma 4 *If $\mathbf{C} = \{(C_j, \omega_j)\}_{j=1}^n$ is an exact fractional cover of $\Gamma = (V,E)$, with $V = [m]$, then*

$$\forall \mathbf{t} \in \mathcal{R}^m, \ \sum_{i=1}^m t_i = \sum_{j=1}^n \omega_j \sum_{k \in C_j} t_k.$$

In particular, $m = \sum_{j=1}^n \omega_j |C_j|$.

3.2 Chromatic PAC-Bayes Bounds

We now provide new PAC-Bayes bounds for classifiers trained on samples $\mathbf{Z}$ drawn from distributions $\mathbf{D}_m$ where dependencies exist. We assume these dependencies are fully determined by $\mathbf{D}_m$ and we define the dependency graph $\Gamma(\mathbf{D}_m)$ of $\mathbf{D}_m$ to be $\Gamma(\mathbf{D}_m) = \Gamma(\mathbf{Z})$. As said before, the marginal distributions of $\mathbf{D}_m$ along each coordinate are the same and are equal to some distribution D.

We introduce additional notation. $\text{PEFC}(\mathbf{D}_m)$ is the set of proper exact fractional covers of $\Gamma(\mathbf{D}_m)$. Given a cover $\mathbf{C} = \{(C_j, \omega_j)\}_{j=1}^n \in \text{PEFC}(\mathbf{D}_m)$, we use the following notation:

- $\mathbf{Z}^{(j)} = \{Z_k\}_{k \in C_j}$;
- $\mathbf{D}_m^{(j)}$, the distribution of $\mathbf{Z}^{(j)}$: it is equal to $D^{|C_j|} = \otimes_{i=1}^{|C_j|} D$ (C_j is independent);
- $\alpha = (\alpha_j)_{1 \leq j \leq n}$ with $\alpha_j = \omega_j / \omega(\mathbf{C})$: we have $\alpha_j \geq 0$ and $\sum_j \alpha_j = 1$;
- $\pi = (\pi_j)_{1 \leq j \leq n}$, with $\pi_j = \omega_j |C_j| / m$: we have $\pi_j \geq 0$ and $\sum_j \pi_j = 1$ (cf. Lemma 4).

In addition, $\mathbf{P}_n$ and $\mathbf{Q}_n$ denote distributions over $\mathcal{H}^n$, P_n^j and Q_n^j are the marginal distributions of $\mathbf{P}_n$ and $\mathbf{Q}_n$ with respect to the jth coordinate, respectively.

We can now state our main results.

Theorem 5 (Chromatic PAC-Bayes Bound (I)) $\forall \mathbf{D}_m,\ \forall \mathcal{H},\ \forall \delta \in (0,1],\ \forall \mathbf{C} = \{(C_j, \omega_j)\}_{j=1}^n \in \text{PEFC}(\mathbf{D}_m),\ \forall \mathbf{P}_n$, *with probability at least* $1-\delta$ *over the random draw of* $\mathbf{Z} \sim \mathbf{D}_m$, *the following holds:*

$$\forall \mathbf{Q}_n,\ \text{kl}(\bar{e}_{\mathbf{Q}_n}(\mathbf{Z}) || e_{\mathbf{Q}_n}) \leq \frac{\omega}{m}\left[\sum_{j=1}^n \alpha_j \text{KL}(Q_n^j || P_n^j) + \ln \frac{m+\omega}{\delta\omega}\right], \tag{5}$$

where ω *stands for* $\omega(\mathbf{C})$, *and*

$$\bar{e}_{\mathbf{Q}_n}(\mathbf{Z}) := \sum_{j=1}^n \pi_j \mathbb{E}_{h\sim Q_n^j} \hat{R}(h, \mathbf{Z}^{(j)}),$$
$$e_{\mathbf{Q}_n} := \mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m} \bar{e}_{\mathbf{Q}_n}(\mathbf{Z}).$$

Proof Deferred to Section 3.4. ■

We would like to emphasize that the same type of result, using the same proof techniques, can be obtained if simple (that is, not exact nor proper) fractional covers are considered. However, as we shall see, the 'best' (in terms of tightness) bound is achieved for covers from the set of proper exact fractional covers, and this is the reason why we have stated Theorem 5 with a restriction to this particular set of covers.

The empirical quantity $\bar{e}_{\mathbf{Q}_n}(\mathbf{Z})$ is a weighted average of the empirical errors on $\mathbf{Z}^{(j)}$ of Gibbs classifiers with respective distributions Q_n^j. The following proposition characterizes $e_{\mathbf{Q}_n} = \mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m} \bar{e}_{\mathbf{Q}_n}(\mathbf{Z})$.

Proposition 6 $\forall \mathbf{D}_m, \forall \mathcal{H}, \forall \mathbf{C} = \{(C_j, \omega_j)\}_{j=1}^n \in \text{PEFC}(\mathbf{D}_m), \forall \mathbf{Q}_n$: $e_{\mathbf{Q}_n} = \mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m} \bar{e}_{\mathbf{Q}_n}(\mathbf{Z})$ *is the error of the Gibbs classifier based on the mixture of distributions* $Q^\pi = \sum_{j=1}^n \pi_j Q_n^j$.

Proof From the definition of π, $\pi_j \geq 0$ and $\sum_{j=1}^n \pi_j = 1$. Thus,

$$\begin{aligned}
\mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m} \bar{e}_{\mathbf{Q}_n}(\mathbf{Z}) &= \mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m} \sum_j \pi_j \mathbb{E}_{h\sim Q_n^j} \hat{R}(h, \mathbf{Z}^{(j)}) \\
&= \sum_j \pi_j \mathbb{E}_{h\sim Q_j} \mathbb{E}_{\mathbf{Z}^{(j)}\sim\mathbf{D}_m^{(j)}} \hat{R}(h, \mathbf{Z}^{(j)}) && \text{(marginalization)} \\
&= \sum_j \pi_j \mathbb{E}_{h\sim Q_n^j} R(h) && (\mathbb{E}_{\mathbf{Z}^{(j)}\sim\mathbf{D}_m^{(j)}} \hat{R}(h, \mathbf{Z}^{(j)}) = R(h), \forall j) \\
&= \mathbb{E}_{h\sim\pi_1 Q_n^1 + \ldots + \pi_j Q_n^j} R(h) = \mathbb{E}_{h\sim Q^\pi} R(h).
\end{aligned}$$

Where, in the third line, we have used the fact that the variables in $\mathbf{Z}^{(j)}$ are identically distributed (by assumption, they are IID). ■

Remark 7 *The prior* $\mathbf{P}_n$ *and the posterior* $\mathbf{Q}_n$ *enter into play in Proposition 6 and Theorem 5 through their marginals only. This advocates for the following learning scheme. Given a cover and a (possibly factorized) prior* $\mathbf{P}_n$, *look for a factorized posterior* $\mathbf{Q}_n = \otimes_{j=1}^n Q_j$ *such that each* Q_j *independently minimizes the usual* IID PAC*-Bayes bound given in Theorem 1 on each* $\mathbf{Z}^{(j)}$. *Then make predictions according to the Gibbs classifier defined with respect to* $Q^\pi = \sum_j \pi_j Q_j$.

The following theorem gives a result that readily applies without choosing a specific cover.

Theorem 8 (Chromatic PAC-Bayes Bound (II)) $\forall \mathbf{D}_m$, $\forall \mathcal{H}$, $\forall \delta \in (0,1]$, $\forall P$, *with probability at least* $1-\delta$ *over the random draw of* $\mathbf{Z} \sim \mathbf{D}_m$, *the following holds*

$$\forall Q,\ \mathrm{kl}(\hat{e}_Q(\mathbf{Z})||e_Q) \leq \frac{\chi^*}{m}\left[\mathrm{KL}(Q||P) + \ln\frac{m+\chi^*}{\delta\chi^*}\right], \tag{6}$$

where χ^* *is the fractional chromatic number of* $\Gamma(\mathbf{D}_m)$, *and where* $\hat{e}_Q(\mathbf{Z})$ *and* e_Q *are as in (2).*

Proof This theorem is just a particular case of Theorem 5. Assume that $\mathbf{C} = \{(C_j, \omega_j)\}_{j=1}^n \in \mathrm{PEFC}(\mathbf{D}_m)$ such that $\omega(C) = \chi^*(\Gamma(\mathbf{D}_m))$, $\mathbf{P}_n = \otimes_{j=1}^n P = P^n$ and $\mathbf{Q}_n = \otimes_{j=1}^n Q = Q^n$, for some P and Q.

For the right-hand side of (6), it directly comes that

$$\sum_j \alpha_j \mathrm{KL}(Q_n^j||P_n^j) = \sum_j \alpha_j \mathrm{KL}(Q||P) = \mathrm{KL}(Q||P).$$

It then suffices to show that $\bar{e}_{\mathbf{Q}_n}(\mathbf{Z}) = \hat{e}_Q(\mathbf{Z})$:

$$\begin{aligned}
\bar{e}_{\mathbf{Q}_n}(\mathbf{Z}) &= \sum_j \pi_j \mathbb{E}_{h\sim Q_n^j} \hat{R}(h, \mathbf{Z}^{(j)}) = \sum_j \pi_j \mathbb{E}_{h\sim Q} \hat{R}(h, \mathbf{Z}^{(j)}) \\
&= \frac{1}{m}\sum_j \omega_j |C_j| \mathbb{E}_{h\sim Q} \frac{1}{|C_j|}\sum_k r(h, Z_k) && (\pi_j = \tfrac{\omega_j|C_j|}{m}, \forall j) \\
&= \mathbb{E}_{h\sim Q} \frac{1}{m}\sum_j \omega_j \sum_k r(h, Z_k) \\
&= \mathbb{E}_{h\sim Q} \frac{1}{m}\sum_i r(h, Z_i) && \text{(cf. Lemma 4)} \\
&= \mathbb{E}_{h\sim Q} \hat{R}(h, \mathbf{Z}) = \hat{e}_Q(\mathbf{Z}).
\end{aligned}$$

■

A few comments are in order.

- A χ^* worsening. This theorem says that even in the case of non IID data, a PAC-Bayes bound very similar to the IID PAC-Bayes bound (1) can be stated, with a worsening (since $\chi^* \geq 1$) proportional to χ^*, that is, proportional to the amount of dependencies in the data. In addition, the new PAC-Bayes bounds is valid with any priors and posteriors, without the need for these distributions to depend on the chosen cover (as is the case with the more general Theorem 5).

- χ^*: the optimal constant. Among all elements of $\mathrm{PEFC}(\mathbf{D}_m)$, χ^* is the best constant achievable in terms of the tightness of the bound (6) on e_Q: getting an optimal coloring gives rise to an 'optimal' bound. Indeed, it suffices to observe that the right-hand side of (5) is decreasing with respect to ω when all Q_n^j are identical (we let the reader check that). As χ^* is the smallest chromatic weight, it gives the tightest bound.

- $\Gamma(\mathbf{D}_m)$ vs. induced subgraphs. If $\mathbf{s} \subseteq [m]$ and $\mathbf{Z}_\mathbf{s} = \{Z_s : s \in \mathbf{s}\}$, it is obvious that Theorem 8 holds for $|\mathbf{s}|$-samples drawn from the marginal distribution $\mathbf{D}_\mathbf{s}$ of $\mathbf{Z}_\mathbf{s}$. Considering only $\mathbf{Z}_\mathbf{s}$ amounts to working with the subgraph $\Gamma(\mathbf{D}_\mathbf{s})$ of $\Gamma(\mathbf{D}_m)$ induced by the vertices in $\mathbf{s}$: this

(a) $\Gamma_{\text{1-edge}}$ (b) Γ_u

Figure 1: Γ_u is the subgraph induced by $\Gamma_{\text{1-edge}}$ (which contains only one edge, between u and v) when u is removed: it might be preferable to consider the distribution corresponding to Γ_u in Theorem 8 instead of the distribution defined wrt $\Gamma_{\text{1-edge}}$, since $\chi^*(\Gamma_{\text{1-edge}}) = 2$ and $\chi^*(\Gamma_u) = 1$ (see text for detailed comments).

might provide a better bound in situations where $\chi^*(\mathbf{D}_\mathbf{s})/|\mathbf{s}|$ is smaller than $\chi^*(\mathbf{D}_m)/m$ (this is not guaranteed, however, because the empirical error $\hat{e}_Q(\mathbf{Z}_\mathbf{s})$ computed on $\mathbf{Z}_\mathbf{s}$ might be larger than $\hat{e}_Q(\mathbf{Z})$). To see this, consider a graph $\Gamma_{\text{1-edge}} = (V,E)$ of m vertices where $|E| = 1$, that is, there are only two nodes, say u and v, that are connected (see Figure 1). The fractional chromatic number $\chi^*_{\text{1-edge}}$ of $\Gamma_{\text{1-edge}}$ is 2 (u and v must use distinct colors) while the (fractional) chromatic number χ^*_u of the subgraph Γ_u of $\Gamma_{\text{1-edge}}$ obtained by removing u is 1: $\chi^*_{\text{1-edge}}$ is twice as big as χ^*_u while the number of nodes only differ by 1 and, for large m, this ratio roughly carries over for $\chi^*_{\text{1-edge}}/m$ and $\chi^*_u/(m-1)$.

This last comment outlines that considering a subset of $\mathbf{Z}$, or, equivalently, a subgraph of $\Gamma(\mathbf{D}_m)$, in (6), might provide a better generalization bound. However, it is assumed that the choice of the subgraph is done *before* computing the bound: the bound does only hold with probability $1-\delta$ for the chosen subgraph. To alleviate this and provide a bound that takes advantage of several induced subgraphs, we have the following proposition:

Proposition 9 *Let* $\{m\}^{\#k}$ *denote* $\{\mathbf{s} : \mathbf{s} \subseteq [m], |\mathbf{s}| = m-k\}$. $\forall \mathbf{D}_m, \forall \mathcal{H}, \forall k \in [m], \forall \delta \in (0,1], \forall P$, *with probability at least* $1-\delta$ *over the random draw of* $\mathbf{Z} \sim \mathbf{D}_m$: $\forall Q$,

$$e_Q \leq \min_{\mathbf{s} \in \{m\}^{\#k}} \left\{ \hat{e}_Q(\mathbf{Z}_\mathbf{s}) + \mathrm{kl}^{-1}\left(\hat{e}_Q(\mathbf{Z}_\mathbf{s}), \frac{\chi^*_\mathbf{s}}{|\mathbf{s}|}\left[\mathrm{KL}(Q||P) + \ln \frac{|\mathbf{s}| + \chi^*_\mathbf{s}}{\chi^*_\mathbf{s}} + \ln \binom{m}{k} + \ln \frac{1}{\delta} \right] \right) \right\}.$$

where $\chi^*_\mathbf{s}$ *is the fractional chromatic number of* $\Gamma(\mathbf{D}_\mathbf{s})$, *and where* $\hat{e}_Q(\mathbf{Z}_\mathbf{s})$ *is the empirical error of the Gibbs classifier* g_Q *on* $\mathbf{Z}_\mathbf{s}$, *that is:* $\hat{e}_Q(\mathbf{Z}_\mathbf{s}) = \mathbb{E}_{h\sim Q}\hat{R}(h, \mathbf{Z}_\mathbf{s})$.

Proof Simply apply the union bound to Equation (6) of Theorem 8: for fixed k, there are $\binom{m}{m-k} = \binom{m}{k}$ subgraphs and using $\delta/\binom{m}{k}$ makes the bound hold with probability $1-\delta$ for all possible $\binom{m}{k}$ subgraphs (simultaneously). Making use of the form (4) gives the result. ■

This bound is particularly useful when, for some small k, there exists a subset $\mathbf{s} \subseteq \{m\}^{\#k}$ such that the induced subgraph $\Gamma(\mathbf{D}_\mathbf{s})$, which has k fewer nodes than $\Gamma(\mathbf{D}_m)$, has a fractional chromatic number $\chi^*_\mathbf{s}$ that is smaller than $\chi^*(\mathbf{D}_m)$ (as is the case with the graph $\Gamma_{\text{1-edge}}$ of Figure 1, where $k = 1$). Obtaining a similar result that holds for subgraphs associated with sets $\mathbf{s}$ of sizes *larger or equal* to $m-k$ is possible by replacing $\ln\binom{m}{k}$ with $\ln\sum_{\kappa=0}^{k}\binom{m}{\kappa}$ in the bound (in that case, k should be kept small enough with respect to m—for example, $k = O_m(1)$—to ensure that the resulting bound still goes down to zero when $m \to \infty$).

3.3 On the Relevance of Fractional Covers

One may wonder whether using the fractional cover framework is the only way to establish a result similar to the one provided by Theorem 5. Of course, this is not the case and one may imagine other ways of deriving closely related results without mentioning the idea of fractional/cover coloring. (For instance, one may manipulate subsets of independent variables, assign weights to these subsets without referring to fractional covers, and arrive at results that are comparable to ours.)

However, if we assume that singling out independent sets of variables is the cornerstone of dealing with interdependent random variables, we find it enlightening to cast our approach within the rich and well-studied fractional cover/coloring framework. On the one hand, our objective of deriving tight bounds amounts to finding a decomposition of the set of random variables at hand into *few and large* independent subsets and taking the graph theory point of view, this obviously corresponds to a problem of graph coloring. Explicitly using the fractional cover/coloring argument allows us to directly benefit from the wealth of related results, such as Property 1 or, for instance, approaches as to how compute a cover or approximate the fractional chromatic number (for instance, linear programming). On the other hand, from a technical point of view, making use of the fractional cover argument allows us to preserve the simple structure of the proof of the classical IID PAC-Bayes bound to derive Theorem 5.

To summarize, the richness of the results on graph (fractional) coloring provides us with elegant tools to deal with a natural representation of the dependencies that may occur within a set of random variables. In addition, and as showed in this article, it is possible to seamlessly take advantage of these tools in the PAC-Bayesian framework (and probably in other bound-related frameworks).

3.4 Proof of Theorem 5

A proof in three steps, following the lines of the proofs given by Seeger (2002a) and Langford (2005) for the IID PAC-Bayes bound, can be provided.

Lemma 10 *$\forall \mathbf{D}_m$, $\forall \delta \in (0,1]$, $\forall \mathbf{C} = \{(C_j, \omega_j)\}_{j=1}^n$, $\forall \mathbf{P}_n$ distribution over $\mathcal{H}^n$, with probability at least $1-\delta$ over the random draw of $\mathbf{Z} \sim \mathbf{D}_m$, the following holds (here, ω stands for $\omega(\mathbf{C})$)*

$$\mathbb{E}_{\mathbf{h}\sim\mathbf{P}_n} \sum_{j=1}^{n} \alpha_j e^{|C_j|\,\mathrm{kl}(\hat{R}(h_j,\mathbf{Z}^{(j)})||R(h_j))} \leq \frac{m+\omega}{\delta\omega},$$

where $\mathbf{h} = (h_1, \ldots, h_n)$ is a random vector of hypotheses.

Proof We first observe the following:

$$\begin{aligned}
\mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m} \sum_j \alpha_j e^{|C_j|\,\mathrm{kl}(\hat{R}(h_j,\mathbf{Z}^{(j)})||R(h_j))} &= \sum_j \alpha_j \mathbb{E}_{\mathbf{Z}^{(j)}\sim\mathbf{D}_m^{(j)}} e^{|C_j|\,\mathrm{kl}(\hat{R}(h,\mathbf{Z}^{(j)})||R(h))} \\
&\leq \sum_j \alpha_j (|C_j|+1) && \text{(Lemma 20, Appendix)} \\
&= \frac{1}{\omega} \sum_j \omega_j (|C_j|+1) \\
&= \frac{m+\omega}{\omega}, && \text{(Lemma 4)}
\end{aligned}$$

where using Lemma 20 is made possible by the fact that $\mathbf{Z}^{(j)}$ is an IID sample. Therefore,

$$\mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m}\mathbb{E}_{\mathbf{h}\sim\mathbf{P}_n}\sum_{j=1}^{n}\alpha_j e^{|C_j|\,\mathrm{kl}(\hat{R}(h_j,\mathbf{Z}^{(j)})||R(h_j))} \leq \frac{m+\omega}{\omega}.$$

According to Markov's inequality (Theorem 22, Appendix),

$$\mathbb{P}_{\mathbf{Z}}\left(\mathbb{E}_{\mathbf{h}\sim\mathbf{P}_n}\sum_{j}\alpha_j e^{|C_j|\,\mathrm{kl}(\hat{R}(h_j,\mathbf{Z}^{(j)})||R(h_j))} \geq \frac{m+\omega}{\omega\delta}\right) \leq \delta.$$

■

Lemma 11 *$\forall\mathbf{D}_m$, $\forall\mathbf{C}=\{(C_j,\omega_j)\}_{j=1}^n$, $\forall\mathbf{P}_n$, $\forall\mathbf{Q}_n$, with probability at least $1-\delta$ over the random draw of $\mathbf{Z}\sim\mathbf{D}_m$, the following holds*

$$\frac{m}{\omega}\sum_{j=1}^{n}\pi_j\mathbb{E}_{h\sim Q_n^j}\mathrm{kl}(\hat{R}(h,\mathbf{Z}^{(j)})||R(h)) \leq \sum_{j=1}^{n}\alpha_j\,\mathrm{KL}(Q_n^j||P_n^j)+\ln\frac{m+\omega}{\delta\omega}.$$

Proof It suffices to use Jensen's inequality (Theorem 21, Appendix) with ln and the fact that $\mathbb{E}_{X\sim P}f(X)=\mathbb{E}_{X\sim Q}\frac{P(X)}{Q(X)}f(X)$, for all f,P,Q. Therefore, $\forall\mathbf{Q}_n$:

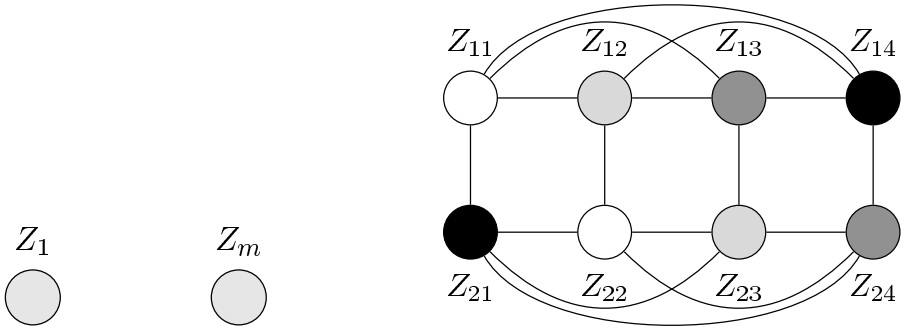
$$\begin{aligned}
\ln\mathbb{E}_{\mathbf{h}\sim\mathbf{P}_n}\sum_{j}\alpha_j e^{|C_j|\,\mathrm{kl}\left(\hat{R}(h_j,\mathbf{Z}^{(j)})||R(h_j)\right)} &= \ln\sum_{j}\alpha_j\mathbb{E}_{h\sim P_n^j}e^{|C_j|\,\mathrm{kl}\left(\hat{R}(h,\mathbf{Z}^{(j)})||R(h)\right)}\\
&= \ln\sum_{j}\alpha_j\mathbb{E}_{h\sim Q_n^j}\frac{P_n^j(h)}{Q_n^j(h)}e^{|C_j|\,\mathrm{kl}\left(\hat{R}(h,\mathbf{Z}^{(j)})||R(h)\right)}\\
&\geq \sum_{j}\alpha_j\mathbb{E}_{h\sim Q_n^j}\ln\left[\frac{P_n^j(h)}{Q_n^j(h)}e^{|C_j|\,\mathrm{kl}\left(\hat{R}(h,\mathbf{Z}^{(j)})||R(h)\right)}\right] \qquad \text{(Jensen's inequality)}\\
&= -\sum_{j}\alpha_j\,\mathrm{KL}(Q_n^j||P_n^j)+\sum_{j}\alpha_j|C_j|\mathbb{E}_{h\sim Q_n^j}\mathrm{kl}\left(\hat{R}(h,\mathbf{Z}^{(j)})||R(h)\right)\\
&= -\sum_{j}\alpha_j\,\mathrm{KL}(Q_n^j||P_n^j)+\frac{m}{\omega}\sum_{j}\pi_j\mathbb{E}_{h\sim Q_n^j}\mathrm{kl}\left(\hat{R}(h,\mathbf{Z}^{(j)})||R(h)\right).
\end{aligned}$$

Lemma 10 then gives the result. ■

Lemma 12 *$\forall\mathbf{D}_m$, $\forall\mathbf{C}=\{(C_j,\omega_j)\}_{j=1}^n$, $\forall\mathbf{Q}_n$,, the following holds*

$$\frac{m}{\omega}\sum_{j=1}^{n}\pi_j\mathbb{E}_{h\sim Q_n^j}\mathrm{kl}(\hat{R}(h,\mathbf{Z}^{(j)})||R(h)) \geq \mathrm{kl}(\bar{e}_Q||e_Q). \tag{7}$$

Proof This simply comes from the convexity of $\mathrm{kl}(x,y)$ in (x,y) (Lemma 23, Appendix). This, in combination with Lemma 11, closes the proof of Theorem 5. ■



(a) IID data (b) Bipartite ranking data

Figure 2: Dependency graphs for different settings described in Section 4. Nodes of the same color are part of the same cover element; hence, they are probabilistically independent. (a) When the data are IID, the dependency graph is disconnected and the fractional number is $\chi^* = 1$; (b) a dependency graph obtained for bipartite ranking from a sample of 4 positive and 2 negative instances: $\chi^* = 4$.

4. Applications

In this section, we provide instances of Theorem 8 for various settings; amazingly, they alllow us to easily derive PAC-Bayes generalization bounds for problems such as ranking and learning from stationary β-mixing processes. The theorems we provide here are all new PAC-Bayes bounds for different non-IID settings.

4.1 IID Case

The first case we are interested in is the IID setting. In this case, the training sample $\mathbf{Z} = \{(X_i, Y_i)\}_{i=1}^m$ is distributed according to $\mathbf{D}_m = D^m$ and the fractional chromatic number of $\Gamma(\mathbf{D}_m)$ is $\chi^* = 1$, since the dependency graph, depicted in Figure 2a is totally disconnected (see Property 1). Plugging in this value of χ^* in the bound of Theorem 8 gives the IID PAC-Bayes bound of Theorem 1. This emphasizes the fact that the standard PAC-Bayes bound is a special case of our more general results.

4.2 General Ranking and Connection to U-Statistics

Here, the learning problem of interest is the following. D is a distribution over $\mathcal{X} \times \mathcal{Y}$ with $\mathcal{Y} = \mathcal{R}$ and one looks for a ranking rule $h \in \mathcal{R}^{\mathcal{X}\times\mathcal{X}}$ that minimizes the *ranking risk* $R^{\text{rank}}(h)$ defined as:

$$R^{\text{rank}}(h) := \mathbb{P}_{\substack{(X,Y)\sim D \\ (X',Y')\sim D}} \left((Y - Y')h(X,X') < 0\right). \tag{8}$$

For a random pair (X,Y), Y can be thought of as a score that allows one to rank objects: given two pairs (X,Y) and (X',Y'), X has a higher rank (or is 'better') than X' if $Y > Y'$. The ranking rule h predicts X to be better than X' if $\text{sign}(h(X,X')) = 1$ and conversely. The objective of learning is to produce a rule h that makes as few misrankings as possible, as measured by (8). Given a finite IID (according to D) sample $\mathbf{S} = \{(X_i, Y_i)\}_{i=1}^{\ell}$ an unbiased estimate of $R^{\text{rank}}(h)$ is $\hat{R}^{\text{rank}}(h, \mathbf{S})$, with:

$$\hat{R}^{\text{rank}}(h, \mathbf{S}) := \frac{1}{\ell(\ell-1)} \sum_{i \neq j} \mathbb{I}_{(Y_i - Y_j)h(X_i,X_j)<0} = \frac{1}{\ell(\ell-1)} \sum_{i \neq j} \mathbb{I}_{Y_{ij}h(X_i,X_j)<0}, \tag{9}$$

where $Y_{ij} := (Y_i - Y_j)$. A natural question is to bound the ranking risk for any learning rule h given $\mathbf{S}$, where the difficulty is that (9) is a sum of identically but not independently random variables, namely the variables $\mathbb{I}_{Y_{ij}h(X_i,X_j)}$.

Let us define $X_{ij} := (X_i, X_j)$, $Z_{ij} := (X_{ij}, Y_{ij})$, and $\mathbf{Z} := \{Z_{ij}\}_{i \neq j}$. We note that the number ℓ of training data suffices to determine the structure of the dependency graph Γ_{rank} of $\mathbf{Z}$ and its distribution, which we denote $\mathbf{D}_{\ell(\ell-1)}$. Henceforth, we are clearly in the framework for the application of the chromatic PAC-Bayes bounds defined in the previous section. In particular, to instantiate Theorem 8 to the present ranking problem, we simply need to have at hand the value χ^*_{rank}, or an upper bound thereof, of the fractional chromatic number of Γ_{rank}. We claim that $\chi^*_{\text{rank}} \leq \ell(\ell-1)/\lfloor \ell/2 \rfloor$ where $\lfloor x \rfloor$ is the largest integer less than or equal to x. We provide the following new PAC-Bayes bound for the ranking risk:

Theorem 13 (Ranking PAC-Bayes bound) *$\forall D$ over $\mathcal{X} \times \mathcal{Y}$, $\forall \mathcal{H} \subseteq \mathcal{R}^{\mathcal{X} \times \mathcal{X}}$, $\forall \delta \in (0,1]$, $\forall P$, with probability at least $1-\delta$ over the random draw of $\mathbf{S} \sim D^\ell$, the following holds*

$$\forall Q,\ \mathrm{kl}(\hat{e}^{rank}_Q(\mathbf{S}) || e^{rank}_Q) \leq \frac{1}{\lfloor \ell/2 \rfloor} \left[\mathrm{KL}(Q||P) + \ln \frac{\lfloor \ell/2 \rfloor + 1}{\delta} \right],$$

where

$$\hat{e}^{rank}_Q(\mathbf{S}) := \mathbb{E}_{h \sim Q} \hat{R}^{rank}(h, \mathbf{S}),$$
$$e^{rank}_Q := \mathbb{E}_{\mathbf{S} \sim D^\ell} \hat{e}^{rank}_Q(\mathbf{S}).$$

Proof We essentially need to prove our claim on the bound on χ^*_{rank}. To do so, we consider a fractional cover of Γ_{rank} motivated by the theory of U-statistics (Hoeffding, 1948, 1963). $\hat{R}(h, \mathbf{S})$ is indeed a U-statistics of order 2 and it might be rewritten as a sum of IID blocks as follows

$$\hat{R}(h, \mathbf{S}) = \frac{1}{\ell(\ell-1)} \sum_{i \neq j} r(h, Z_{ij}) = \frac{1}{\ell!} \sum_{\sigma \in \Sigma_\ell} \frac{1}{\lfloor \ell/2 \rfloor} \sum_{i=1}^{\lfloor \ell/2 \rfloor} r\left(h, Z_{\sigma(i)\sigma(\lfloor \ell/2 \rfloor + i)}\right),$$

where Σ_ℓ is the set of permutations over $[\ell]$. The innermost sum is obviously a sum of IID random variables as no two summands share the same indices.

A proper exact fractional cover $\mathbf{C}_{\text{rank}}$ can be derived from this decomposition as[1]

$$\mathbf{C}_{\text{rank}} := \left\{ \left(C_\sigma := \left\{ Z_{\sigma(i)\sigma(\lfloor \ell/2 \rfloor + i)} \right\}_{i=1}^{\lfloor \ell/2 \rfloor}, \omega_\sigma := \frac{1}{(\ell-2)! \lfloor \ell/2 \rfloor} \right) \right\}_{\sigma \in \Sigma_\ell}.$$

Indeed, as remarked before, each C_σ is an independent set and each random variable Z_{pq} for $p \neq q$, appears in exactly $(\ell-2)! \times \lfloor \ell/2 \rfloor$ sets C_σ (for i fixed, the number of permutations σ such that $\sigma(i) = p$ and $\sigma(\lfloor \ell/2 \rfloor + i) = q$ is equal to $(\ell-2)!$, that is, the number of permutations on $\ell - 2$ elements; as i can take $\lfloor \ell/2 \rfloor$ values, this gives the result). Therefore, $\forall p, q, p \neq q$:

$$\sum_{\sigma \in \Sigma_\ell} \omega_\sigma \mathbb{I}_{Z_{pq} \in C_\sigma} = \frac{1}{(\ell-2)! \lfloor \ell/2 \rfloor} \sum_{\sigma \in \Sigma_\ell} \mathbb{I}_{Z_{pq} \in C_\sigma} = \frac{1}{(\ell-2)! \lfloor \ell/2 \rfloor} \times (\ell-2)! \lfloor \ell/2 \rfloor = 1,$$

1. Note that the cover defined here considers elements C_σ containing random variables themselves instead of their indices. This abuse of notation is made for sake of readability.

which proves that $\mathbf{C}_{\text{rank}}$ is a proper exact fractional cover. Its weight $\omega(\mathbf{C}_{\text{rank}})$ is

$$\omega(\mathbf{C}_{\text{rank}}) = \ell! \times \omega_\sigma = \frac{\ell(\ell-1))}{\lfloor \ell/2 \rfloor}.$$

Hence, from the definition of χ^*_{rank},

$$\chi^*_{\text{rank}} \leq \frac{\ell(\ell-1))}{\lfloor \ell/2 \rfloor}.$$

The theorem follows by an instantiation of Theorem 8 with $m := \ell(\ell-1)$ and the bound on χ^*_{rank} we have just proven. ■

To our knowledge, this is the first PAC-Bayes bound on the ranking risk, while a Rademacher-complexity based analysis was given by Clémençon et al. (2008). In the proof, we have used arguments from the analysis of U-processes, which allow us to easily derive a convenient fractional cover of the dependency graph of $\mathbf{Z}$. Note however that our framework still applies even if not all the Z_{ij}'s are known, as required if an analysis based on U-processes is undertaken. This is particularly handy in practical situations where one may only be given the values Y_{ij}—but *not* the values of Y_i and Y_j—for a limited number of (i, j) pairs (and not all the pairs).

An interesting question is to know how the so-called Hoeffding decomposition used by Clémençon et al. (2008) to establish fast rates of convergence for empirical ranking risk minimizers could be used to draw possibly tighter PAC-Bayes bounds. This would imply being able to appropriately take advantage of moments of order 2 in PAC-Bayes bounds, and a possible direction for that has been proposed by Lacasse et al. (2006). This is left for future work as it is not central to the present paper.

Of course, the ranking rule may be based on a scoring function $f \in \mathcal{R}^{\mathcal{X}}$ such that $h(X,X') = f(X) - f(X')$, in which case all the results that we state in terms of h can be stated similarly in terms of f. This is important to note from a practical point of view as it is probably more usual to learn functions defined over $\mathcal{X}$ rather than $\mathcal{X} \times \mathcal{X}$ (as is h).

Finally, we would like to stress that the bound on χ^*_{rank} that we have exhibited is actually rather tight. Indeed, it is straightforward to see that the clique number of Γ_{rank} is $2(\ell-1)$ (the cliques are made of variables $\{Z_{ip}\}_p \bigcup \{Z_{pi}\}_p$ for every i), and according to Property 1, $2(\ell-1)$ is therefore a lower bound on χ^*_{rank}. If ℓ is even, then our bound on χ^*_{rank} is equal to $2(\ell-1)$ and so is χ^*_{rank}; if ℓ is odd, then our bound is 2ℓ.

4.3 Bipartite Ranking and a Bound on the AUC

A particular ranking setting is that of bipartite ranking, where $\mathcal{Y} = \{-1, +1\}$. Let D be a distribution over $\mathcal{X} \times \mathcal{Y}$ and D_{+1} (D_{-1}) be the class conditional distribution $D_{X|Y=+1}$ ($D_{X|Y=-1}$) with respect to D. In this setting (see, for instance, Agarwal et al., 2005), one may be interested in controlling what we call the *bipartite misranking risk* $R^{\text{AUC}}(h)$ (the reason for the AUC superscript will become clear in the sequel), of a ranking rule $h \in \mathcal{R}^{\mathcal{X}\times\mathcal{X}}$ by

$$R^{\text{AUC}}(h) := \mathbb{P}_{\substack{X\sim D_{+1} \\ X'\sim D_{-1}}} (h(X,X') < 0). \tag{10}$$

Note that the relation between R^{AUC} and R^{rank} (cf. Equation 8) can be made clear whenever the hypotheses h under consideration are such that $h(x,x')$ and $h(x',x)$ have opposite signs. In this situation, it is straightforward to see that

$$R^{\text{rank}}(h) = 2\mathbb{P}(Y=+1)\mathbb{P}(Y=-1)R^{\text{AUC}}(h).$$

Let $\mathbf{S} = \{(X_i, Y_i)\}_{i=1}^{\ell}$ be an IID sample distributed according to $\mathbf{D}_\ell = D^\ell$. The empirical bipartite ranking risk $\hat{R}^{\text{AUC}}(h, \mathbf{S})$ of h on $\mathbf{S}$ defined as

$$\hat{R}^{\text{AUC}}(h, \mathbf{S}) := \frac{1}{\ell^+ \ell^-} \sum_{\substack{i:Y_i=+1 \\ j:Y_j=-1}} \mathbb{I}_{h(X_i,X_j)<0} \tag{11}$$

where ℓ^+ (ℓ^-) is the number of positive (negative) data in $\mathbf{S}$, estimates the fraction of pairs (X_i, X_j) that are incorrectly ranked incorrectly (given that $Y_i = +1$ and $Y_j = -1$) by h: it is an unbiased estimator of $R^{\text{AUC}}(h)$.

As before, h may be expressed in terms of a scoring function $f \in \mathcal{R}^{\mathcal{X}}$ such that $h(X, X') = f(X) - f(X')$, in which case (overloading notation):

$$R^{\text{AUC}}(f) = \mathbb{P}_{\substack{X \sim D_{+1} \\ X' \sim D_{-1}}} (f(X) < f(X')) \text{ and } \hat{R}^{\text{AUC}}(f, \mathbf{S}) = \frac{1}{\ell^+ \ell^-} \sum_{\substack{i:Y_i=+1 \\ j:Y_j=-1}} \mathbb{I}_{f(X_i)<f(X_j)},$$

where we recognize in $\hat{R}^{\text{AUC}}(f, \mathbf{S})$ one minus the Area under the ROC curve, or AUC, of f on $\mathbf{S}$ (Agarwal et al., 2005; Cortes and Mohri, 2004), hence the AUC superscript in the name of the risk. As a consequence, providing a PAC-Bayes bound on $R^{\text{AUC}}(h)$ (or $R^{\text{AUC}}(f)$) amounts to providing a generalization (lower) bound on the AUC, which is a widely used measure in practice to evaluate the performance of a scoring function.

Let us define $X_{ij} := (X_i, X_j)$, $Z_{ij} := (X_{ij}, 1)$ and $\mathbf{Z} := \{Z_{ij}\}_{ij:Y_i=+1, Y_j=-1}$, that is, $\mathbf{Z}$ is a sequence of pairs X_{ij} made of one positive example and one negative example. We then are once again in the framework defined earlier,[2] that is, the Z_{ij}'s share the same distribution but are dependent on each other, since Z_{ij} depends on $\{Z_{pq} : p = i \text{ or } q = j\}$ (see Figure 2). Note that in order to ease the reading of the present subsection, we make the implicit decomposition of training set $\mathbf{S}$ into $\mathbf{S} = \mathbf{S}^+ \cup \mathbf{S}^-$, where $\mathbf{S}^+$ (resp. $\mathbf{S}^-$) is made of the ℓ^+ (ℓ^-) positive (negative) data of $\mathbf{S}$; the size ℓ of $\mathbf{S}$ is therefore $\ell = \ell^+ + \ell^-$. This decomposition entails a separate reindexing of the positive (negative) data from 1 to ℓ^+ (from 1 to ℓ^-).

Building on Theorem 8, we have the following result:

Theorem 14 (AUC PAC-Bayes bound) *$\forall D$ over $\mathcal{X} \times \mathcal{Y}$, $\forall \mathcal{H} \subseteq \mathcal{R}^{\mathcal{X} \times \mathcal{X}}$, $\forall \delta \in (0, 1]$, $\forall P$, with probability at least $1 - \delta$ over the random draw of $\mathbf{S} \sim D^\ell$, the following holds*

$$\forall Q,\ \text{kl}(\hat{e}_Q^{\text{AUC}}(\mathbf{S}) || e_Q^{\text{AUC}}) \leq \frac{1}{\ell_{\min}} \left[\text{KL}(Q||P) + \ln \frac{\ell_{\min} + 1}{\delta} \right],$$

where $\ell_{\min} = \min(\ell^+, \ell^-)$, and

$$\hat{e}_Q^{\text{AUC}}(\mathbf{S}) := \mathbb{E}_{h \sim Q} \hat{R}^{\text{AUC}}(h, \mathbf{S}),$$
$$e_Q^{\text{AUC}} := \mathbb{E}_{\mathbf{S} \sim D^\ell} \hat{e}_Q^{\text{AUC}}(\mathbf{S}).$$

Proof The proof works in three steps and borrows ideas from Agarwal et al. (2005). The first two parts are necessary to deal with the fact that the dependency graph of $\mathbf{Z}$, as it depends on the random sample $\mathbf{S}$, does not have a deterministic structure.

2. The slight difference with what has been described above is that the dependency graph is now a random variable: it depends on the Y_i's. It is shown in the proof of Theorem 14 how this can be dealt with.

4.3.1 CONDITIONING ON $\mathbf{Y} = \mathbf{y}$

Let $\mathbf{y} \in \{-1,+1\}^\ell$ be a fixed vector and let $\ell^+_\mathbf{y}$ and $\ell^-_\mathbf{y}$ be the number of positive and negative labels in $\mathbf{y}$, respectively. We define the distribution $\mathbf{D}_\mathbf{y}$ as $\mathbf{D}_\mathbf{y} := \otimes_{i=1}^{\ell} D_{y_i}$; this is a distribution on $\mathcal{X}^\ell$. With a slight abuse of notation, $\mathbf{D}_\mathbf{y}$ will also be used to denote the distribution over $(\mathcal{X}\times\mathcal{Y})^\ell$ of samples $\mathbf{S} = \{(X_i,y_i)\}_{i=1}^{\ell}$ such that the sequence $\{X_i\}_{i=1}^{\ell}$ is distributed according to $\mathbf{D}_\mathbf{y}$. It is easy to check that $\forall h \in \mathcal{H}, \mathbb{E}_{\mathbf{S}\sim\mathbf{D}_\mathbf{y}} \hat{R}^{\text{rank}}(h,\mathbf{S}) = R^{\text{rank}}(h)$ (cf. Equations 10 and 11).

Given $\mathbf{S}$, if we define, as said earlier, $X_{ij} := (X_i, X_j)$, $Y_{ij} := 1$ and $Z_{ij} := (X_{ij}, Y_{ij})$, then $\mathbf{Z} := \{Z_{ij}\}_{i:y_i=1,j:y_j=-1}$ is a sample of identically distributed variables, each with distribution $D_{\pm 1} = D_{+1} \otimes D_{-1} \otimes \mathbf{1}$ over $\mathcal{X}\times\mathcal{X}\times\mathcal{Y}$, where $\mathcal{Y} = \{-1,+1\}$ and where $\mathbf{1}$ is the distribution that produces 1 with probability 1.

Letting $m = \ell^+_\mathbf{y}\ell^-_\mathbf{y}$ we denote by $\mathbf{D}_{\mathbf{y},m}$ the distribution of the training sample $\mathbf{Z}$, within which interdependencies exist, as illustrated in Figure 2. Theorem 8 can thus be directly applied to classifiers trained on $\mathbf{Z}$, the structure of $\Gamma(\mathbf{D}_{\mathbf{y},m})$ and its corresponding fractional chromatic number $\chi^*_\mathbf{y}$ being completely determined by $\mathbf{y}$. Hence, letting $\mathcal{H} \subseteq \mathcal{R}^{\mathcal{X}\times\mathcal{X}}$, we have: $\forall \delta \in (0,1], \forall P$ over $\mathcal{H}$, with probability at least $1-\delta$ over the random draw of $\mathbf{Z} \sim \mathbf{D}_{\mathbf{y},m}$,

$$\forall Q,\ \mathrm{kl}(\hat{e}_Q(\mathbf{Z})||e_Q) \leq \frac{\chi^*_\mathbf{y}}{m}\left[\mathrm{KL}(Q||P) + \ln\frac{m+\chi^*_\mathbf{y}}{\delta\chi^*_\mathbf{y}}\right],$$

where $\hat{e}_Q(\mathbf{Z}) = \mathbb{E}_{h\sim Q}\hat{R}(h,\mathbf{Z}) = \mathbb{E}_{h\sim Q}\sum_{ij}\mathbb{I}_{Y_{ij}h(Z_{ij})<0} = \mathbb{E}_{h\sim Q}\sum_{ij}\mathbb{I}_{h(Z_{ij})<0}$, which is exactly equal to $\hat{e}^{\text{AUC}}_Q(\mathbf{S})$ (cf. Equation 11); likewise, $e_Q = \mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_{\mathbf{y},m}}\hat{e}_Q(\mathbf{Z}) = \mathbb{E}_{\mathbf{S}\sim\mathbf{D}_\mathbf{y}}\hat{e}^{\text{AUC}}_Q(\mathbf{S}) = e^{\text{AUC}}_Q$. Hence, $\forall \delta \in (0,1], \forall P$, with probability at least $1-\delta$ over the random draw of $\mathbf{S} \sim \mathbf{D}_\mathbf{y}$,

$$\forall Q,\ \mathrm{kl}(\hat{e}^{\text{AUC}}_Q(\mathbf{S})||e^{\text{AUC}}_Q) \leq \frac{\chi^*_\mathbf{y}}{m}\left[\mathrm{KL}(Q||P) + \ln\frac{m+\chi^*_\mathbf{y}}{\delta\chi^*_\mathbf{y}}\right]. \tag{12}$$

4.3.2 UNCONDITIONING ON $\mathbf{Y}$

As proposed by Agarwal et al. (2005), let us call $\Phi(P,\mathbf{S},\delta)$ the event (12); we just stated that $\forall \mathbf{y} \in \{-1,+1\}^\ell, \forall P, \forall \delta \in (0,1], \mathbb{P}_{\mathbf{S}\sim\mathbf{D}_\mathbf{y}}(\Phi(P,\mathbf{S},\delta)) \geq 1-\delta$, or, equivalently

$$\mathbb{P}_{\mathbf{S}\sim\mathbf{D}_\ell}(\neg\Phi(P,\mathbf{S},\delta)|Y=\mathbf{y}) = \mathbb{P}_{\mathbf{S}\sim\mathbf{D}_\mathbf{y}}(\neg\Phi(P,\mathbf{S},\delta)) < \delta,$$

that is to say, the conditional (to $Y=\mathbf{y}$) probability of the event $\neg\Phi(P,\mathbf{S},\delta)$ is bounded by δ. This directly implies that the unconditional probability of $\neg\Phi(P,\mathbf{S},\delta)$ is bounded by δ as well:

$$\mathbb{P}_{\mathbf{S}\sim\mathbf{D}_\ell}(\neg\Phi(P,\mathbf{S},\delta)) \leq \mathbb{P}_{\mathbf{S}\sim\mathbf{D}_\ell}(\neg\Phi(P,\mathbf{S},\delta)|Y=\mathbf{y}) < \delta.$$

Hence, $\forall \delta \in (0,1], \forall P$, with probability at least $1-\delta$ over the random draw of $\mathbf{S} \sim \mathbf{D}_\ell$,

$$\forall Q,\ \mathrm{kl}(\hat{e}^{\text{AUC}}_Q||e^{\text{AUC}}_Q) \leq \frac{\chi^*_\mathbf{S}}{m_\mathbf{S}}\left[\mathrm{KL}(Q||P) + \ln\frac{m_\mathbf{S}+\chi^*_\mathbf{S}}{\delta\chi^*_\mathbf{S}}\right]. \tag{13}$$

where $\chi^*_\mathbf{S}$ is the fractional chromatic number of the graph $\Gamma(\mathbf{Z})$, with $\mathbf{Z}$ defined from $\mathbf{S}$ as in the first part of the proof, where the observed (random) labels are now taken into account; here $m_\mathbf{S} = \ell^+\ell^-$, where ℓ^+ (ℓ^-) is the number of positive (negative) data in $\mathbf{S}$.

4.3.3 COMPUTING THE FRACTIONAL CHROMATIC NUMBER

In order to finish the proof, it suffices to observe that, for $\mathbf{Z} = \{Z_{ij}\}_{ij}$, if $\ell_{\max} = \max(\ell^+, \ell^-)$, then the fractional chromatic number of $\Gamma(\mathbf{Z})$ is $\chi^* = \ell_{\max}$.

Indeed, the clique number of $\Gamma(\mathbf{Z})$ is $\ell_{\max}$ as for all $i = 1, \ldots, \ell^+$ ($j = 1, \ldots, \ell^-$), $\{Z_{ij} : j = 1, \ldots, \ell^-\}$ ($\{Z_{ij} : i = 1, \ldots, \ell^+\}$) defines a clique of order ℓ^- (ℓ^+) in $\Gamma(\mathbf{Z})$. Thus, from Property 1: $\chi \geq \chi^* \geq \ell_{\max}$.

A proper exact cover $\mathbf{C} = \{C_k\}_{k=1}^{\ell_{\max}}$ of $\Gamma(\mathbf{Z})$ can be constructed as follows. Suppose that $\ell_{\max} = \ell^+$, then $C_k = \{Z_{i\sigma_k(i)} : i = 1, \ldots, \ell^-\}$, with

$$\sigma_k(i) = (i + k - 2 \mod \ell^+) + 1,$$

is an independent set: no two variables Z_{ij} and Z_{pq} in C_k are such that $i = p$ or $j = q$. In addition, it is straightforward to check that $\mathbf{C}$ is indeed a cover of $\Gamma(\mathbf{Z})$. This cover is of size $\ell^+ = \ell_{\max}$, which means that it achieves the minimal possible weight over proper exact (fractional) covers since $\chi^* \geq \ell_{\max}$. Hence, $\chi^* = \chi = \ell_{\max} (= c(\Gamma))$. Plugging in this value of χ^* in (13), and noting that $m_{\mathbf{S}} = \ell_{\max}\ell_{\min}$ with $\ell_{\min} = \min(\ell^+, \ell^-)$, closes the proof. ■

We observe that in the theorem, the dependence on the skew of the sample is expressed in terms of $1/\min(\ell^+, \ell^-)$, whereas in the the works of Agarwal et al. (2005) and Usunier et al. (2005), the bound depends on the larger $1/\ell^+ + 1/\ell^-$.

The PAC-Bayes bound of Theorem 14 can be specialized to the case where $h(x, x') = f(x) - f(x')$ with $f \in \{x \mapsto w \cdot x : w \in \mathcal{X}\}$: f is therefore a linear scoring function and $h(x, x') = w \cdot (x - x')$. The ranking rule h is thus a linear classifier acting on the difference of its arguments (the next result we present therefore carries over to kernel classifiers). As proposed by Langford (2005), we may assume an isotropic Gaussian prior $P = \mathcal{N}(0, I)$ and a family of posteriors $Q_{w,\mu}$ parameterized by $w \in \overline{\mathcal{X}}$ and $\mu > 0$ such that $Q_{w,\mu}$ is $\mathcal{N}(\mu, 1)$ in the direction w and $\mathcal{N}(0, 1)$ in all perpendicular directions, we arrive at the following theorem:

Theorem 15 (AUC Linear PAC-Bayes bound) *$\forall \ell, \forall D$ over $\mathcal{X} \times \mathcal{Y}$, $\forall \delta \in (0, 1]$, the following holds with probability at least $1 - \delta$ over the draw of $\mathbf{S} \sim D^\ell$:*

$$\forall w, \mu > 0,\ \mathrm{kl}(\hat{e}^{\mathrm{AUC}}_{Q_{w,\mu}}(\mathbf{S}) || e^{\mathrm{AUC}}_{Q_{w,\mu}}) \leq \frac{1}{\ell_{\min}} \left[\frac{\mu^2}{2} + \ln \frac{\ell_{\min} + 1}{\delta} \right].$$

Proof Straightforward from the bound of Langford (2005) and Theorem 14. ■

Note that this specific parameterization of Q could have been done in Theorem 13 as well. We arbitrarily choose to provide it for this AUC based bound as learning linear ranking rule by AUC minimization is a common approach (Ataman et al., 2006; Brefeld and Scheffer, 2005; Rakotomamonjy, 2004), and the presented result may be of practical interest (for model selection purpose, for instance) for a larger audience.

The bounds given in Theorem 14 and Theorem 15 are very similar to what we would get if applying IID PAC-Bayes bound to one (independent) element C_j of a minimal cover (i.e., its weight equals the fractional chromatic number) $\mathbf{C} = \{C_j\}_{j=1}^n$ such as the one we used in the proof of Theorem 14. This would imply the empirical error $\hat{e}^{\mathrm{rank}}_{Q}$ to be computed on only one specific C_j and not all the C_j's simultaneously, as is the case for the new results. It turns out that, for proper exact fractional covers $\mathbf{C} = \{(C_j, \omega)\}_{j=1}^n$ with elements C_j having the same size, it is better, in terms of

absolute moments of the empirical error, to assess it on the whole data set, rather than on only one C_j. The following proposition formalizes this.

Proposition 16 $\forall \mathbf{D}_m, \forall \mathcal{H}, \forall \mathbf{C} = \{(C_j, \omega_j\}_{j=1}^n \in \text{PEFC}(\mathbf{D}_m), \forall Q, \forall r \in \mathcal{N}, r \geq 1$, *if* $|C_1| = \ldots = |C_n|$ *then*

$$\mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m}|\hat{e}_Q(\mathbf{Z}) - e_Q|^r \leq \mathbb{E}_{\mathbf{Z}^{(j)}\sim\mathbf{D}_m^{(j)}}|\hat{e}_Q(\mathbf{Z}^{(j)}) - e_Q|^r, \forall j \in \{1, \ldots n\}.$$

Proof Using the convexity of $|\cdot|^r$ for $r \geq 1$, the linearity of $\mathbb{E}$ and the notation of Section 3, for $\mathbf{Z} \sim \mathbf{D}_m$:

$$\begin{aligned}
|\hat{e}_Q(\mathbf{Z}) - e_Q|^r &= |\sum_j \pi_j (\mathbb{E}_{h\sim Q}\hat{R}(h, \mathbf{Z}^{(j)}) - R(h))|^r \\
&\leq \sum_j \pi_j |\mathbb{E}_{h\sim Q}(\hat{R}(h, \mathbf{Z}^{(j)}) - R(h))|^r \\
&= \sum_j \pi_j |\hat{e}_Q(\mathbf{Z}^{(j)}) - e_Q|^r.
\end{aligned}$$

Taking the expectation of both sides with respect to $\mathbf{Z}$ and noting that the random variables $|\hat{e}_Q(\mathbf{Z}^{(j)}) - e_Q|^r$, have the same distribution, gives the result. ■

This proposition upholds the idea of Pemmaraju (2001) to base the decomposition of a dependency graph on equitable coloring.

4.4 β-mixing Processes

Here, we provide a PAC-Bayes theorem for classifiers trained on data from a stationary β-mixing process, of which we recall some definitions, as formulated by Yu (1994) (see also, for example, Mohri and Rostamizadeh, 2009).

Definition 17 (Stationarity) *A sequence of random variables* $\mathbf{Z} = \{Z_t\}_{t=-\infty}^{+\infty}$ *is* stationary *if, for any t and nonnegative integer m and k, the random subsequences* $(Z_t, \ldots, Z_{t+m})$ *and* $(Z_{t+k}, \ldots, Z_{t+m+k})$ *are identically distributed.*

Definition 18 (β-mixing process) *Let* $\mathbf{Z} = \{Z_t\}_{t=-\infty}^{+\infty}$ *be a stationary sequence of random variables. For any* $i, j \in \mathcal{Z} \cup \{-\infty, +\infty\}$, *let* σ_i^j *denote the* σ*-algebra generated by the random variables* Z_k, $i \leq k \leq j$. *Then, for any positive integer k, the* β*-mixing coefficient* $\beta(k)$ *of the stochastic process* $\mathbf{Z}$ *is defined as*

$$\beta(k) = \sup_{n\geq 1} \mathbb{E} \sup \left\{ |\mathbb{P}(A|\sigma_1^n) - \mathbb{P}(A)| : A \in \sigma_{n+k}^{+\infty} \right\}.$$

$\mathbf{Z}$ *is said to be* β*-mixing if* $\beta(k) \to 0$ *when* $k \to \infty$.

(Note there is an equivalent definition of the β-mixing coefficient based on finite partitions; see Yu, 1994 for details.) Stationary β-mixing processes model a situation where the interdependence between the random variables at hand is temporal. When the process is mixing, it means that the strengh of dependence between variables weakens over times.

The bound that we propose is in the same vein as the one proposed by Mohri and Rostamizadeh (2009), with the difference that our bound is a PAC-Bayes bound and theirs a Rademacher-complexity-based bounds. In addition to being a new type of data-dependent bound for the case of stationary

β-mixing process, we may anticipate that, in practical situations, our bound inherits the tightness of the IID PAC-Bayes bound (whereas, to the best of our knowledge, there is no evidence of such practicality for Rademacher-complexity-based bounds).

Let us state our generalization bound for classifiers trained on samples $\mathbf{Z}$ drawn from stationary β-mixing distributions.

Theorem 19 (β-mixing process PAC-Bayes bound) *Let m be a positive integer. Let $\mathbf{D}^{\beta}$ be a stationary β-mixing distribution over $\mathcal{Z}$ and $\mathbf{D}_m^{\beta}$ be the distribution of m-samples according to $\mathbf{D}^{\beta}$. $\forall \mathcal{H} \subseteq \mathcal{R}^{\mathcal{X}}$, $\forall \mu, a \in \mathcal{N}$ such that $2\mu a = m$, $\forall \delta \in (2(\mu-1)\beta(a), 1]$, $\forall P$, with probability at least $1-\delta$ over the random draw of $\mathbf{Z} \sim \mathbf{D}_m^{\beta}$, the following holds*

$$\forall Q,\ \mathrm{kl}(\hat{e}_Q^{\beta}(\mathbf{Z}) || e_Q^{\beta}) \leq \frac{1}{\mu}\left[\mathrm{KL}(Q||P) + \ln \frac{2(\mu+1)}{\delta - 2(\mu-1)\beta(a)}\right],$$

where

$$\hat{e}_Q^{\beta}(\mathbf{Z}) := \mathbb{E}_{h\sim Q}\hat{R}(h,\mathbf{Z}) = \mathbb{E}_{h\sim Q}\sum_{t=1}^{m} \mathbb{I}_{Y_t h(X_t)<0},$$
$$e_Q^{\beta} := \mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m^{\beta}}\hat{e}_Q^{\beta}(\mathbf{Z}).$$

Proof The proof makes use of the independent block decomposition proposed by Yu (1994), our chromatic PAC-Bayes bound of Theorem 8, and Corollary 24 (Appendix).

4.4.1 THE CHROMATIC BOUND FOR INDEPENDENT BLOCKS

Let $\mathbf{Z} = \{Z_1, \ldots, Z_m\}$ be the random variables we have to deal with. If μ and a are two integers such that $2\mu a = m$ (we assume that m is even, if it is odd one may drop the last variable Z_m and work on a sample of size $m-1$). Then $\mathbf{Z}$ can be decomposed into two subsequences $\mathbf{Z}_0$ and $\mathbf{Z}_1$ as follows:

$$\mathbf{Z}_0 := \{\mathbf{Z}_0^s := (Z_{a(2s-2)+1}, \ldots, Z_{a(2s-2)+a}) : s \in [\mu]\},$$
$$\mathbf{Z}_1 := \{\mathbf{Z}_1^s := (Z_{a(2s-1)+1}, \ldots, Z_{a(2s-1)+a}) : s \in [\mu]\}.$$

Both $\mathbf{Z}_0$ and $\mathbf{Z}_1$ are made of μ blocks of a consecutive random variables. The blocks are interdependent as well as the variables within each block. $\mathbf{D}_0$ will denote the distribution of $\mathbf{Z}_0$.

We now define a sequence $\underline{\mathbf{Z}}$ of independent blocks as:

$$\underline{\mathbf{Z}} := \{\underline{\mathbf{Z}}^s := (Z_1^s, \ldots, Z_a^s) : s \in [\mu]\},$$

such that the blocks $\underline{\mathbf{Z}}^s$ are mutually independent and such that each block $\underline{\mathbf{Z}}^s$ has the same distribution as $\mathbf{Z}_0^s$, that is, from the stationarity assumption, the distribution of $\mathbf{Z}_0^1$ (the blocks $\underline{\mathbf{Z}}^s$ are IID).

The dependency graph $\underline{\Gamma}$ of $\underline{\mathbf{Z}}$ is such that all the variables in a block are all connected and such that there are no connections between blocks. Theorem 8 can readily be applied to the random sample $\underline{\mathbf{Z}}$, whose distribution we denote $\underline{\mathbf{D}}$: for all P and $\delta \in (0,1]$,

$$\mathbb{P}_{\underline{\mathbf{Z}}\sim\underline{\mathbf{D}}}\left(\Phi(P,\underline{\mathbf{Z}},\delta)\right) < \delta, \tag{14}$$

with $e_Q := \mathbb{E}_{\underline{\mathbf{Z}}\sim\underline{\mathbf{D}}}\hat{e}_Q(\underline{\mathbf{Z}})$ and $\Phi(P,\mathbf{Z},\delta)$ is the event defined as:

$$\Phi(P,\mathbf{Z},\delta) := \left\{ \exists Q,\ \mathrm{kl}(\hat{e}_Q(\mathbf{Z})||e_Q) > \frac{1}{\mu}\left[\mathrm{KL}(Q||P) + \ln\frac{\mu+1}{\delta}\right]\right\}.$$

To see why and how Theorem 8 can be used to get (14), observe that:

- the number of variables in $\underline{\mathbf{Z}}$ is μa;
- by stationarity, all variables Z_α^s, for $\alpha \in [a]$ and $s \in [\mu]$ share the same distribution: we therefore do actually work with dependent but identically distributed variables;
- the (fractional) chromatic number $\underline{\chi}^*$ of $\underline{\Gamma}$ is a, since
  1. the clique number is a (that is, the number of variables in each block),
  2. the cover $\underline{\mathbf{C}}$ of $\underline{\Gamma}$ with

$$\underline{\mathbf{C}} := \left\{ \left(C_\alpha := \{Z_\alpha^1,\ldots,Z_\alpha^\mu\}, 1\right)\right\}_{1\le\alpha\le a}$$

  is a proper exact cover of size a.

Noting that, consequently

$$\frac{\underline{\chi}^*}{\mu a} = \frac{a}{\mu a} = \frac{1}{\mu} \qquad \text{and} \qquad \frac{\mu a + \underline{\chi}^*}{\delta\underline{\chi}^*} = \frac{\mu a + a}{\delta a} = \frac{\mu+1}{\delta},$$

gives the expression of $\Phi(P,\mathbf{Z},\delta)$ and (14).

The last two steps of the proof are similar to those used by Mohri and Rostamizadeh (2009) to establish their bound.

4.4.2 A BOUND FOR $\mathbf{Z}_0$

To establish the bound for $\mathbf{Z}_0$, it suffices to use Corollary 24 (Appendix) with $c(\mathbf{z})$ being defined as:

$$c(\mathbf{z}) := \mathbb{I}_{\Phi(P,\mathbf{z},\delta)},$$

which is a bounded measurable function on the blocks $\mathbf{Z}_0^s$ (and thus on the blocks $\underline{\mathbf{Z}}_s$). We have:

$$|\mathbb{E}_{\mathbf{Z}_0\sim\mathbf{D}_0} c(\mathbf{Z}_0) - \mathbb{E}_{\underline{\mathbf{Z}}\sim\underline{\mathbf{D}}} c(\underline{\mathbf{Z}})| \le (\mu-1)\beta(a),$$

and therefore, since $\mathbb{P}_{\mathbf{Z}_0\sim\mathbf{D}_0}(\Phi(P,\mathbf{Z}_0,\delta)) = \mathbb{E}_{\mathbf{Z}_0\sim\mathbf{D}_0} c(\mathbf{Z}_0)$ and $\mathbb{P}_{\underline{\mathbf{Z}}\sim\underline{\mathbf{D}}}(\Phi(P,\underline{\mathbf{Z}},\delta)) = \mathbb{E}_{\underline{\mathbf{Z}}\sim\underline{\mathbf{D}}} c(\underline{\mathbf{Z}})$:

$$\begin{aligned}\mathbb{P}_{\mathbf{Z}_0\sim\mathbf{D}_0}(\Phi(P,\mathbf{Z}_0,\delta)) &\le \mathbb{P}_{\underline{\mathbf{Z}}\sim\underline{\mathbf{D}}}(\Phi(P,\underline{\mathbf{Z}},\delta)) + (\mu-1)\beta(a) && (15)\\ &< \delta + (\mu-1)\beta(a). && \text{(cf. Equation 14)}\end{aligned}$$

4.4.3 ESTABLISHING THE BOUND

Finally, observe that:

$$\begin{aligned}
\Phi(P,\mathbf{Z},\delta) &\Rightarrow \exists Q: \frac{1}{2}\mathrm{kl}(\hat{e}_Q(\mathbf{Z}_0)||e_Q)+\frac{1}{2}\mathrm{kl}(\hat{e}_Q(\mathbf{Z}_1)||e_Q) > \frac{1}{\mu}\left[\mathrm{KL}(Q||P)+\ln\frac{\mu+1}{\delta}\right] \\
&\Rightarrow \exists Q: \bigvee_{i\in\{0,1\}}\left\{\mathrm{kl}(\hat{e}_Q(\mathbf{Z}_i)||e_Q)) > \frac{1}{\mu}\left[\mathrm{KL}(Q||P)+\ln\frac{\mu+1}{\delta}\right]\right\} \\
&\Rightarrow \bigvee_{i\in\{0,1\}}\left\{\exists Q: \mathrm{kl}(\hat{e}_Q(\mathbf{Z}_i)||e_Q)) > \frac{1}{\mu}\left[\mathrm{KL}(Q||P)+\ln\frac{\mu+1}{\delta}\right]\right\} \\
&\Leftrightarrow \Phi(P,\mathbf{Z}_0,\delta)\vee\Phi(P,\mathbf{Z}_1,\delta),
\end{aligned}$$

where we used $\hat{e}_Q(\mathbf{Z}) = \hat{e}_Q(\mathbf{Z}_0)/2+\hat{e}_Q(\mathbf{Z}_1)/2$ and the convexity of kl in the first line.

This leads to:

$$\begin{aligned}
\mathbb{P}_{\mathbf{Z}\sim\mathbf{D}_m^\beta}(\Phi(P,\mathbf{Z},\delta)) &\leq \mathbb{P}_{\mathbf{Z}\sim\mathbf{D}_m^\beta}(\Phi(P,\mathbf{Z}_0,\delta)\vee\Phi(P,\mathbf{Z}_1,\delta)) \\
&\leq \mathbb{P}_{\mathbf{Z}\sim\mathbf{D}_m^\beta}(\Phi(P,\mathbf{Z}_0,\delta))+\mathbb{P}_{\mathbf{Z}\sim\mathbf{D}_m^\beta}(\Phi(P,\mathbf{Z}_1,\delta)) && \text{(union bound)} \\
&= 2\mathbb{P}_{\mathbf{Z}\sim\mathbf{D}_m^\beta}(\Phi(P,\mathbf{Z}_0,\delta)) && \text{(stationarity)} \\
&= 2\mathbb{P}_{\mathbf{Z}_0\sim\mathbf{D}_0}(\Phi(P,\mathbf{Z}_0,\delta)) && \text{(marginalization wrt } \mathbf{Z}_0) \\
&\leq 2\delta+2(\mu-1)\beta(a). && \text{(cf. Equation 15)}
\end{aligned}$$

Adjusting δ to $\delta/2-(\mu-1)\beta(a)$ ends the proof. ■

5. Conclusion

In this work, we propose the first PAC-Bayes bounds applying for classifiers trained on non-IID data. The derivation of these results rely on the use of fractional covers of graphs, convexity and standard tools from probability theory. The results that we provide are very general and can easily be instantiated for specific learning settings such as ranking and learning from from mixing distributions: amazingly, we obtain at a very low cost original PAC-Bayes bounds for these settings. Using a generalized PAC-Bayes bound, we provide in the appendix a chromatic PAC-Bayes bound that holds for non-independently and non-identically distributed data: it allows us to derive a PAC-Bayes bound for classifiers trained on data from a stationary φ-mixing distribution.

This work gives rise to many interesting questions. First, it seems that using a fractional cover to decompose the non-IID training data into sets of IID data and then tightening the bound through the use of the chromatic number is some form of variational relaxation as often encountered in the context of inference in graphical models, the graphical model under consideration in this work being one that encodes the dependencies in $\mathbf{D}_m$. It might be interesting to make this connection clearer to see if, for instance, tighter and still general bounds can be obtained with more appropriate variational relaxations than the one incurred by the use of fractional covers.

Besides, Theorem 5 advocates for the learning algorithm described in Remark 7. We would like to see how such a learning algorithm based on possibly multiple priors/multiple posteriors could perform empirically and how tight the proposed bound could be.

On another empirical side, it might be interesting to run simulations on bipartite ranking problems to see how accurate the bound of Theorem 15 can be: we expect the results to be of good quality, because of the resemblance of the bound of the theorem with the IID PAC-Bayes theorem for margin classifiers, which has proven to be rather accurate (Langford, 2005). The work of Germain et al. (2009) is also another contribution that tends to support that a practical use of our bounds should provide competitive results (note that Theorem 25 gives a sufficient condition for the general PAC-Bayes bound of Germain et al. (2009) to be non degenerate). Likewise, it would be interesting to see how the possibly more accurate PAC-Bayes bound for large margin classifiers proposed by Langford and Shawe-taylor (2002), which should translate to the case of bipartite ranking as well, performs empirically. The question also remains as to what kind of strategies to learn the prior(s) could be used to render the bound of Theorem 5 the tightest possible. This is one of the most stimulating question as performing such prior learning makes it possible to obtain very accurate generalization bound (Ambroladze et al., 2007).

The connection between our ranking bounds and the theory of U-statistics makes it possible to envision the use of higher order moments in establishing PAC-Bayes bounds, thanks to Hoeffding's decomposition. We plan to investigate further in this direction, for both the ranking measures we have studied (noting that the AUC is a two-sample U-statistics Hoeffding, 1963).

Finally, we have been working on a more general way to establish chromatic bounds from IID bounds (covering VC, Rademacher, PAC-Bayes and—possibly—binomial tail test set bounds), without the need to perform 'low-level' calculations such as the ones proposed in Section 3.4. The meta-bound that we have been developing is in the spirit of that proposed by Blanchard and Fleuret (2007), except that the randomization we propose is on the subsets constituting the fractional cover (and not the hypothesis set). In other terms, given a cover $\mathbf{C} = \{(C_j, \omega_j)\}_j$, the fact that an IID bound holds on one subset C_j of a cover is considered as a random event, the probability of a subset to be chosen being $\omega_j/\omega(\mathbf{C})$. A simple union bound gives our generic result, which translates into cover-independent (but fractional-chromatic-number-dependent) chromatic bounds such as (6) (Theorem 8) under very mild conditions on the shape of the base IID bound. Along with that work, we try to answer the question of establishing a principled way to handle situations where random variables show weak dependencies (as is the case for β-mixing processes), as for now, the framework described here applies when variables are either dependent or independent, disregarding the magnitude of the dependencies—our PAC-Bayes bound for β-mixing processes would then be a specific case of such general result.

Acknowledgments

This work is partially supported by the IST Program of the EC, under the FP7 Pascal 2 Network of Excellence, ICT-216886-NOE, and by ANR projects LAMPADA and SEQUOIA.

Appendix A. Technical Lemmas

This appendix gathers useful (and well-known) results for the different proofs.

Lemma 20 *Let D be a distribution over $\mathcal{Z}$.*

$$\forall h \in \mathcal{H}, \mathbb{E}_{\mathbf{Z}\sim D^m} e^{m\,\mathrm{kl}\left(\hat{R}(h,\mathbf{Z})||R(h)\right)} \leq m+1.$$

Proof Let $h \in \mathcal{H}$. For $\mathbf{z} \in \mathcal{Z}^m$, we let $q(\mathbf{z}) = \hat{R}(h,\mathbf{z})$; we also let $p = R(h)$. Note that since $\mathbf{Z}$ is i.i.d, $mq(\mathbf{Z})$ is binomial with parameters m and p (recall that $r(h,Z)$ takes the values 0 and 1 upon correct and erroneous classification of Z by h, respectively).

$$\begin{aligned}
\mathbb{E}_{\mathbf{Z}\sim D^m} e^{m\,\mathrm{kl}(q(\mathbf{Z})||p)} &= \sum_{\mathbf{z}\in\mathcal{Z}^m} e^{m\,\mathrm{kl}(q(\mathbf{z})||p)} \mathbb{P}_{\mathbf{Z}\sim D^m}(\mathbf{Z}=\mathbf{z}) \\
&= \sum_{0\le k\le m} e^{m\,\mathrm{kl}\left(\frac{k}{m}||p\right)} \mathbb{P}_{\mathbf{Z}\sim D^m}(mq(\mathbf{Z})=k) \\
&= \sum_{0\le k\le m} \binom{m}{k} e^{m\,\mathrm{kl}\left(\frac{k}{m}||p\right)} p^k (1-p)^{m-k} \\
&= \sum_{0\le k\le m} \binom{m}{k} e^{m\left(\frac{k}{m}\ln\frac{k}{m} + (1-\frac{k}{m})\ln(1-\frac{k}{m})\right)} \\
&= \sum_{0\le k\le m} \binom{m}{k} \left(\frac{k}{m}\right)^k \left(1-\frac{k}{m}\right)^{m-k}.
\end{aligned}$$

However, it is obvious that, from the definition of the binomial distribution,

$$\forall m \in \mathcal{N}, \forall k \in [0,m], \forall t \in [0,1], \binom{m}{k} t^k (1-t)^{m-k} \le 1.$$

This is obviously the case for $t = \frac{k}{m}$, which gives

$$\sum_{0\le k\le m} \binom{m}{k} \left(\frac{k}{m}\right)^k \left(1-\frac{k}{m}\right)^{m-k} \le \sum_{0\le k\le m} 1 = m+1.$$

■

Theorem 21 (Jensen's inequality) *Let $f \in \mathcal{R}^{\mathcal{X}}$ be a convex function. For all probability distribution P on $\mathcal{X}$:*

$$f(\mathbb{E}_{X\sim P} X) \le \mathbb{E}_{X\sim P} f(X).$$

Theorem 22 (Markov's Inequality) *Let X be a positive random variable on $\mathcal{R}$, such that $\mathbb{E}X < \infty$.*

$$\forall t \in \mathcal{R}, \mathbb{P}_X\left\{X \ge \frac{\mathbb{E}X}{t}\right\} \le \frac{1}{t}.$$

Consequently: $\forall M \ge \mathbb{E}X, \forall t \in \mathcal{R}, \mathbb{P}_X\left\{X \ge \frac{M}{t}\right\} \le \frac{1}{t}$.

Lemma 23 (Convexity of kl) $\forall p,q,r,s \in [0,1], \forall \alpha \in [0,1]$,

$$\mathrm{kl}(\alpha p + (1-\alpha)q || \alpha r + (1-\alpha)s) \le \alpha\,\mathrm{kl}(p||r) + (1-\alpha)\,\mathrm{kl}(q||s).$$

Proof It suffices to see that $f \in \mathcal{R}^{[0,1]^2}, f(\mathbf{v} = [p\ q]) = \mathrm{kl}(q||p)$ is convex over $[0,1]^2$: the Hessian H of f is

$$H = \begin{bmatrix} \frac{q}{p^2} + \frac{1-q}{(1-p)^2} & -\frac{1}{p} - \frac{1}{1-p} \\ -\frac{1}{p} - \frac{1}{1-p} & \frac{1}{q} + \frac{1}{1-q} \end{bmatrix},$$

and, for $p,q \in [0,1]$, $\frac{q}{p^2} + \frac{1-q}{(1-p)^2} \geq 0$ and $\det H = \frac{(p-q)^2}{q(1-q)p^2(1-p)^2} \geq 0$: $H \succeq 0$ and f is indeed convex. ■

Finally, we have the following version by Mohri and Rostamizadeh (2009) of Corollary 2.7 in Yu (1994), which is based on the definition of the blocks $\mathbf{Z}_k^s$:

Corollary 24 *Let c be a measurable function defined with respect to the blocks $\mathbf{Z}_0^s$. If c has absolute value bounded by M, then*

$$|\mathbb{E}_{\mathbf{Z}_0 \sim \mathbf{D}_0} c(\mathbf{Z}) - \mathbb{E}_{\underline{\mathbf{Z}} \sim \underline{\mathbf{D}}} c(\underline{\mathbf{Z}})| \leq (\mu - 1) M \beta(a).$$

Appendix B. Applications of a Generic PAC-Bayes Theorem

Let us first recall the following generic PAC-Bayes result, which is a corollary/compound of results proposed by Seeger (2002b) and McAllester (2003). In particular, the γ function need not be differentiable with respect to its second argument and it applies to any 'risk' functional ψ for which a concentration inequality exists.

Corollary 25 (Generic PAC-Bayes Theorem) *Let $\mathcal{H} \subseteq \mathcal{R}^{\mathcal{X}}$ and $\psi : \mathcal{H} \times \bigcup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{R}$. If there exist $\alpha \geq 1, \beta > 1$ and a nonnegative convex function $\Delta : \mathcal{R} \times \mathcal{R} \to \mathcal{R}_+$ that is strictly increasing with respect to its second argument such that*

$$\forall h \in \mathcal{H}, \forall \varepsilon > 0,\ \mathbb{P}_{\mathbf{Z} \sim \mathbf{D}_m} \left[\mathbb{E}\psi(h) - \psi(h, \mathbf{Z}) \geq \varepsilon \right] \leq \alpha \exp\left(-\beta \Delta(\mathbb{E}\psi(h), \varepsilon) \right), \tag{16}$$

where $\mathbb{E}\psi(h)$ stands for $\mathbb{E}_{\mathbf{Z} \sim \mathbf{D}_m} \psi(h, \mathbf{Z})$, then, $\forall P$, with probability at least $1 - \delta$ over the draw of $\mathbf{Z} \sim \mathbf{D}_m$:

$$\forall Q,\ \Delta(e_Q^{\psi}, e_Q^{\psi} - \hat{e}_Q^{\psi}(\mathbf{Z})) \leq \frac{1}{\beta - 1} \left[\mathrm{KL}(Q||P) + \ln \frac{\alpha\beta}{\delta} \right].$$

where

$$\hat{e}_Q^{\psi}(\mathbf{Z}) := \mathbb{E}_{h \sim Q} \psi(h, \mathbf{Z})$$
$$e_Q^{\psi} := \mathbb{E}_{\mathbf{Z}} \hat{e}_Q^{\psi}(\mathbf{Z}) = \mathbb{E}_{h \sim Q} \mathbb{E}_{\mathbf{Z}} \psi(h, \mathbf{Z})$$

Proof Along lines from Seeger (2002b) and McAllester (2003).

1. Observe that, thanks to Lemma 26 (below) with $\delta(\varepsilon) := \Delta(\mathbb{E}\psi(h), \varepsilon)$,

   $$\mathbb{E}_{\mathbf{Z}} e^{(\beta-1)\Delta(\mathbb{E}\psi(h), \mathbb{E}\psi(h) - \psi(h,\mathbf{Z}))} \leq \alpha\beta, \text{ and, } \mathbb{E}_{h \sim P} \mathbb{E}_{\mathbf{Z}} e^{(\beta-1)\Delta(\mathbb{E}\psi(h), \mathbb{E}\psi(h) - \psi(h,\mathbf{Z}))} \leq \alpha\beta$$

   Applying Markov's inequality then gives:

   $$\mathbb{P}_{\mathbf{Z}} \left[\mathbb{E}_{h \sim P} e^{(\beta-1)\Delta(\mathbb{E}\psi(h), \mathbb{E}\psi(h) - \psi(h,\mathbf{Z}))} \geq \frac{\alpha\beta}{\delta} \right] \leq \delta$$

2. Using the entropy extremal inequality $\ln \mathbb{E}_{X \sim P} f(X) \geq -\mathrm{KL}(Q||P)) + \mathbb{E}_{X \sim Q} \ln f(X)$, $\forall P, Q, X$ (see the proof of Lemma 11), and the fact that $x \mapsto \ln x$ is nondecreasing, the previous step leads to

   $$\mathbb{P}_{\mathbf{Z}} \left[\exists Q : -\mathrm{KL}(Q||P) + (\beta - 1) \mathbb{E}_{h \sim Q} \Delta(\mathbb{E}\psi(h), \mathbb{E}\psi(h) - \psi(h, \mathbf{Z})) \geq \ln \frac{\alpha\beta}{\delta} \right] \leq \delta.$$

3. Since Δ is convex, Jensen's inequality can be used to give (here, $h \sim Q$)

$$\mathbb{P}_{\mathbf{Z}}\left[\exists Q : -\mathrm{KL}(Q||P) + (\beta-1)\Delta(\mathbb{E}_{h,\mathbf{Z}}\psi(h,\mathbf{Z}), \mathbb{E}_{h,\mathbf{Z}}\psi(h,\mathbf{Z}) - \mathbb{E}_h\psi(h,\mathbf{Z})) \geq \ln\frac{\alpha\beta}{\delta}\right] \leq \delta.$$

■

Lemma 26 (McAllester, 2003) *Let X be a real-valued random variable on $\mathcal{X}$ and $\alpha \geq 1, \beta > 1$. Let $\delta : \mathcal{R} \to \mathcal{R}$ be a nonnegative and strictly increasing function. We have:*

$$\forall x \in \mathcal{R}, \mathbb{P}[X \geq x] \leq \alpha e^{-\beta\delta(x)} \Rightarrow \mathbb{E}\left[e^{(\beta-1)\delta(X)}\right] \leq \alpha\beta.$$

Proof See the proof of McAllester (2003). Here, we take α into account. As f is strictly increasing:

$$\mathbb{P}[X \geq x] = \mathbb{P}[\delta(X) \geq \delta(x)] = \mathbb{P}\left[e^{(\beta-1)\delta(X)} \geq e^{(\beta-1)\delta(x)}\right].$$

Hence: $\mathbb{P}\left[e^{(\beta-1)\delta(X)} \geq e^{(\beta-1)\delta(x)}\right] \leq \alpha e^{-\beta\delta(x)}$. Setting $\nu = e^{(\beta-1)\delta(x)}$, we get:

$$\mathbb{P}\left[e^{(\beta-1)\delta(X)} \geq \nu\right] \leq \min(1, \alpha\nu^{-\beta/(\beta-1)}).$$

Thus, as for a nonnegative random variable W, $\mathbb{E}[W] = \int_0^\infty \mathbb{P}[W \geq \nu]d\nu$:

$$\mathbb{E}\left[e^{(\beta-1)\delta(X)}\right] \leq 1 + \alpha\int_1^\infty \nu^{-\beta/(\beta-1)} = 1 + \alpha(\beta-1).$$

Since $\alpha > 1$, $1 + \alpha(\beta-1) \leq \alpha\beta$, which ends the proof. ■

We observe that:

- if $\psi(h,\mathbf{Z}) = \sum_{i=1}^m \mathbb{I}_{Y_i h(Xi)<0}$ then, by the one-sided Chernoff bound, $\alpha = 1, \beta = m$ and $\Delta(p,\varepsilon) = \mathrm{kl}(p-\varepsilon||p)$ make Equation (16) hold. The PAC-Bayes bound provided by Corollary 25 is that of Theorem 1 where m is replaced by $m-1$;

- if

$$\forall i \in [m], \sup_{z_1,\ldots,z_m,z_i' \in \mathcal{Z}} |\psi(z_1,\ldots,z_m) - \psi(z_1,\ldots,z_{i-1},z_i',z_{i+1},\ldots,z_m)| \leq c_i,$$

  then, thanks to McDiarmid inequality (McDiarmid, 1989), $\alpha = 1, \beta = 2/\sum_i c_i^2$ and $\Delta(p,\varepsilon) = \varepsilon^2$, make Equation (16) hold and a PAC-Bayes bound can be derived (we let the reader write the corresponding PAC-Bayes bound);

- it suffices to have an appropriate concentration inequality for the problem at hand to have an effective PAC-Bayes bound.

B.1 Generalized Chromatic PAC-Bayes Bound

To get a chromatic PAC-Bayes theorem for non-identically non-independently distributed data, we simply make use of the following concentration inequality of Janson (2004).

Theorem 27 (Janson, 2004) *Suppose that $\mathbf{Z} = \{Z_i\}_{i=1}^m$ is an m-sample of real-valued random variables distributed according to some distribution $\mathbf{D}_m$. Suppose that each Z_i has range $[a_i, b_i]$. If $S_\mathbf{Z} = \sum_{i=1}^m Z_i$, then,*

$$\forall \varepsilon > 0,\ \mathbb{P}_{S_\mathbf{Z}}\left[\mathbb{E}S_\mathbf{Z} - S_\mathbf{Z} \geq \varepsilon\right] \leq \exp\left[-\frac{2\varepsilon^2}{\chi^*(\mathbf{D}_m)\sum_{i=1}^m (b_i - a_i)^2}\right],$$

where $\chi^*(\mathbf{D}_m)$ is the fractional chromatic number of the dependency graph of $\mathbf{D}_m$.

Note that *no assumption* is made on the Z_i's being identically distributed.

This concentration inequality gives rise to the following generalized chromatic PAC-Bayes bound that applies to non independently, possibly non identically distributed data and allows us to use any bounded loss functions r.

Theorem 28 (Generalized Chromatic PAC-Bayes Bound) *$\forall \mathbf{D}_m$, $\forall \mathcal{H}$, $\forall \delta \in (0,1]$, $\forall P$, with probability at least $1-\delta$ over the random draw of $\mathbf{Z} \sim \mathbf{D}_m$, the following holds*

$$\forall Q,\ |\hat{e}_Q(\mathbf{Z}) - e_Q|^2 \leq \frac{\chi^* M^2}{2m - \chi^* M^2}\left[\mathrm{KL}(Q||P) + \ln\frac{2m}{\chi^* M^2} + \ln\frac{1}{\delta}\right],$$

where χ^* stands for $\chi^*(\mathbf{D}_m)$, r is a bounded function with range M and

$$\hat{e}_Q(\mathbf{Z}) := \mathbb{E}_{h\sim Q}\hat{R}(h,\mathbf{Z})$$
$$e_Q := \mathbb{E}_{h\sim Q}\hat{e}_Q(\mathbf{Z}) = \mathbb{E}_{h\sim Q}\mathbb{E}_{\mathbf{Z}\sim\mathbf{D}_m}\hat{R}(h,\mathbf{Z}),$$

with $\hat{R}(h,\mathbf{Z}) := \sum_i r(h,Z_i)/m$.

Proof It suffices to apply Corollary 25 with Theorem 27, $\alpha = 1$, $\Delta(p,\varepsilon) = \varepsilon^2$ and $\beta = 2m/\chi^* M$ (since, as r has range M, $\hat{R}$ has range M/m). ■

We notice the following.

- Here, as no assumption is done regarding the identical distribution of the Z_i's, the expected risk $R(h) = \mathbb{E}_\mathbf{Z}\hat{R}(h,\mathbf{Z})$ does not unfold as in (3).
- In the case of using identically distributed random variables and the 0-1 loss, there is no concentration inequality that allows us to retrieve the tighter PAC-Bayes bound given in Theorem 8.
- From a more general point of view, it is enticing to try to establish even more generic results resting on the principle of graph coloring with the aim of decoupling this approach to the PAC-Bayesian framework. This is the subject of ongoing work.

B.2 φ-mixing PAC-Bayes Bound

The definition of a φ-mixing process follows.

Definition 29 (φ-mixing process) *Let $\mathbf{Z} = \{Z_t\}_{t=-\infty}^{+\infty}$ be a stationary sequence of random variables. For any $i, j \in \mathbb{Z} \cup \{-\infty, +\infty\}$, let σ_i^j denote the σ-algebra generated by the random variables Z_k, $i \leq k \leq j$. Then, for any positive integer k, the φ-mixing coefficient $\varphi(k)$ of the stochastic process $\mathbf{Z}$ is defined as*

$$\varphi(k) = \sup_{n, A \in \sigma_{n+k}^{+\infty}, B \in \sigma_{-\infty}^{n}} \left| \mathbb{P}[A|B] - \mathbb{P}[A] \right|.$$

$\mathbf{Z}$ is said to be φ-mixing if $\varphi(k) \to 0$ as $k \to 0$.

In order to establish our new PAC-Bayes bounds for stationary φ-mixing distributions, it suffices to make use of the following concentration inequality by Kontorovich and Ramanan (2008).

Theorem 30 (Kontorovich and Ramanan, 2008) *Let $\psi : \mathcal{U}^m \to \mathcal{R}$ be a function defined over a countable space $\mathcal{U}$. If ψ is l-Lipschitz with respect to the Hamming metric for some $l > 0$, then the following holds for all $t > 0$:*

$$\mathbb{P}_{\mathbf{Z}}\left[|\psi(\mathbf{Z}) - \mathbb{E}_{\mathbf{Z}}[\psi(\mathbf{Z})]| > t \right] \leq 2 \exp\left[-\frac{t^2}{2ml^2 \|\Lambda_m\|_\infty^2} \right],$$

where $\|\Lambda_m\|_\infty \leq 1 + 2\sum_{k=1}^{m} \varphi(k)$.

Suppose that the loss function r is again such that it takes values in $[0, M]$. Then, for any $h \in \mathcal{H}$, the function $\psi(\mathbf{Z}) = \frac{1}{m}\sum_{i=1}^{m} r(h, Z_i) = \hat{R}(h, \mathbf{Z})$ is obviously M/m-Lipschitz. Therefore, for a sample $\mathbf{Z}$ drawn according to a φ-mixing process, we have the following concentration inequality on $\hat{R}(h, \mathbf{Z})$ that holds for any $h \in \mathcal{H}$:

$$\mathbb{P}_{\mathbf{Z} \sim \mathbf{D}_m}\left[\left| \hat{R}(h, \mathbf{Z}) - R(h) \right| > t \right] \leq 2 \exp\left[-\frac{mt^2}{2M^2 \|\Lambda_m\|_\infty^2} \right]. \tag{17}$$

We directly get the following PAC-Bayes bound for φ-mixing processes.

Theorem 31 (PAC-Bayes bound for stationary φ-mixing processes) *Let $\mathbf{D}^{\varphi}$ be a stationary φ-mixing distribution over $\mathcal{Z}$ and $\mathbf{D}_m^{\varphi}$ be the distribution of m-samples according to $\mathbf{D}^{\varphi}$. $\forall \mathcal{H} \subseteq \mathcal{R}^{\mathcal{X}}$, $\forall \delta \in (0, 1]$, $\forall P$, with probability at least $1 - \delta$ over the random draw of $\mathbf{Z} \sim \mathbf{D}_m^{\varphi}$, the following holds*

$$\forall Q,\ |\hat{e}_Q^{\varphi}(\mathbf{Z}) - e_Q^{\varphi}|^2 \leq \frac{2M^2 \|\Lambda_m\|_\infty^2}{m - 2M^2 \|\Lambda_m\|_\infty^2} \left[\mathrm{KL}(Q||P) + \ln \frac{m}{M^2 \|\Lambda_m\|_\infty^2} + \ln \frac{1}{\delta} \right],$$

where $\|\Lambda_m\|_\infty \leq 1 + 2\sum_{k=1}^{m} \varphi(k)$, $r(h, Z) = \mathbb{I}_{Yh(X)<0}$ and

$$\hat{e}_Q^{\varphi}(\mathbf{Z}) := \mathbb{E}_{h \sim Q} \hat{R}(h, \mathbf{Z}) = \mathbb{E}_{h \sim Q} \sum_{t=1}^{m} \mathbb{I}_{Y_t h(X_t)<0}$$

$$e_Q^{\varphi} := \mathbb{E}_{\mathbf{Z} \sim \mathbf{D}_m^{\varphi}} \hat{e}_Q^{\varphi}(\mathbf{Z}).$$

Proof Equation (17), and Corollary 25 with $\alpha = 2$, $\beta = m/(2M^2\|\Lambda\|_\infty^2)$, $\Delta(p, \varepsilon) = \varepsilon^2$. ■

References

S. Agarwal and P. Niyogi. Generalization bounds for ranking algorithms via algorithmic stability. *Journal of Machine Learning Research*, 10:441–474, 2009.

S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

A. Ambroladze, E. Parrado-Hernandez, and J. Shawe-Taylor. Tighter PAC-Bayes bounds. In *Adv. in Neural Information Processing Systems 19*, pages 9–16, 2007.

K. Ataman, W. Nick, and Y. Zhang. Learning to rank by maximizing auc with linear programming. In *In IEEE International Joint Conference on Neural Networks (IJCNN 2006)*, pages 123–129, 2006.

J.-Y. Audibert and O. Bousquet. Combining PAC-bayesian and generic chaining bounds. *Journal of Machine Learning Research*, 8:863–889, 2007. ISSN 1533-7928.

P. Bartlett, O. Bousquet, and S. Mendelson. Local rademacher complexities. *Annals of Statistics*, 33(4):1497–1537, 2005.

P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

G. Blanchard and F. Fleuret. Occam's hammer. In *COLT*, pages 112–126, 2007.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, March 2002.

U. Brefeld and T. Scheffer. AUC maximizing support vector learning. In *Proc. of the ICML Workshop on ROC Analysis in Machine Learning*, 2005.

O. Catoni. *PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning*, volume 56 of *Lecture Notes–Monograph Series*. Beachwood, Ohio, USA: Institute of Mathematical Statistics, 2007.

S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and empirical minimization of U-statistics. *The Annals of Statistics*, 36(2):844–874, April 2008. ISSN 0090-5364.

C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *Adv. in Neural Information Processing Systems 16*, 2004.

Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

P. Germain, A. Lacasse, F. Laviolette, and M. Marchand. PAC-Bayesian learning of linear classifiers. In *Proc. of the 26th Annual International Conference on Machine Learning*, pages 353–360, 2009.

R. Herbrich and T. Graepel. A PAC-Bayesian margin bound for linear classifiers: Why svms work. In *Advances in Neural Information Processing Systems 13*, pages 224–230, 2001.

W. Hoeffding. A class of statistics with asymptotically normal distribution. *Annals of Mathematical Statistics*, 19(3):293–325, 1948.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

S. Janson. Large deviations for sums of partly dependent random variables. *Random Structures Algorithms*, 24:234–248, 2004.

L. Kontorovich and K. Ramanan. Concentration inequalities for dependent random variables via the martingale method. *The Annals of Probability*, 36(6):2126–2158, 2008.

A. Lacasse, F. Laviolette, M. Marchand, P. Germain, and N. Usunier. PAC-Bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier. In *Advances in Neural Information Processing Systems 19*, pages 769–776, 2006.

J. Langford. Tutorial on Practical Theory for Classification. *Journal of Machine Learning Research*, pages 273–306, 2005.

J. Langford and J. Shawe-taylor. PAC-Bayes and margins. In *Adv. in Neural Information Processing Systems 15*, pages 439–446, 2002.

D. McAllester. Some PAC-Bayesian Theorems. *Machine Learning*, 37:355–363, 1999.

D. McAllester. Simplified pac-bayesian margin bounds. In *Proc. of the 16th Annual Conference on Computational Learning Theory*, pages 203–215, 2003.

C. McDiarmid. On the method of bounded differences. *Survey in Combinatorics*, pages 148–188, 1989.

M. Mohri and A. Rostamizadeh. Rademacher complexity bounds for non-i.i.d. processes. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1097–1104, 2009.

S. V. Pemmaraju. Equitable coloring extends chernoff-hoeffding bounds. In *RANDOM-APPROX*, pages 285–296, 2001.

A. Rakotomamonjy. Optimizing the area under the ROC curve with SVMs. In *ROC Analysis in Artificial Intelligence*, pages 71–80, 2004.

E.R. Schreinerman and D.H. Ullman. *Fractional Graph Theory: A Rational Approach to the Theory of Graphs*. Wiley Interscience Series in Discrete Math., 1997.

M. Seeger. PAC-Bayesian generalization bounds for Gaussian processes. *Journal of Machine Learning Research*, 3:233–269, 2002a.

M. Seeger. The proof of McAllester's PAC-Bayesian theorem. Technical report, Institute for ANC, Edinburgh, UK, 2002b.

N. Usunier, M.-R. Amini, and P. Gallinari. A data-dependent generalisation error bound for the AUC. In *Proc. of the ICML Workshop on ROC Analysis in Machine Learning*, 2005.

N. Usunier, M.-R. Amini, and P. Gallinari. Generalization error bounds for classifiers trained with interdependent data. In *Adv. in Neural Information Processing Systems 18*, pages 1369–1376, 2006.

B. Yu. Rates of convergence for empirical processes of stationary mixing sequences. *Annals of Probability*, 22(1):94–116, 1994.

Journal of Machine Learning Research 11 (2010) 1957-2000 Submitted 6/08; Revised 11/09; Published 7/10

Practical Approaches to Principal Component Analysis in the Presence of Missing Values

Alexander Ilin ALEXANDER.ILIN@TKK.FI
Tapani Raiko TAPANI.RAIKO@TKK.FI
Adaptive Informatics Research Center
Aalto University School of Science and Technology
P.O. Box 15400, FI-00076 Aalto, Finland

Editor: Tommi Jaakkola

Abstract

Principal component analysis (PCA) is a classical data analysis technique that finds linear transformations of data that retain the maximal amount of variance. We study a case where some of the data values are missing, and show that this problem has many features which are usually associated with nonlinear models, such as overfitting and bad locally optimal solutions. A probabilistic formulation of PCA provides a good foundation for handling missing values, and we provide formulas for doing that. In case of high dimensional and very sparse data, overfitting becomes a severe problem and traditional algorithms for PCA are very slow. We introduce a novel fast algorithm and extend it to variational Bayesian learning. Different versions of PCA are compared in artificial experiments, demonstrating the effects of regularization and modeling of posterior variance. The scalability of the proposed algorithm is demonstrated by applying it to the Netflix problem.

Keywords: principal component analysis, missing values, overfitting, regularization, variational Bayes

1. Introduction

Principal component analysis (PCA) is a data analysis technique that can be traced back to Pearson (1901). It can be used to compress data sets of high dimensional vectors into lower dimensional ones. This is useful, for instance, in visualization and feature extraction. PCA has been extensively covered in the literature (e.g., Jolliffe, 2002; Bishop, 2006; Diamantaras and Kung, 1996; Haykin, 1989; Cichocki and Amari, 2002). PCA can be derived from a number of starting points and optimization criteria. The most important of these are minimization of the mean-square error in data compression, finding mutually orthogonal directions in the data having maximal variances, and decorrelation of the data using orthogonal transformations.

In the data compression formulation, PCA finds a smaller-dimensional linear representation of data vectors such that the original data could be reconstructed from the compressed representation with the minimum square error. Assume that we have n $d \times 1$ data vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ that are modeled as

$$\mathbf{y}_j \approx \mathbf{W}\mathbf{x}_j + \mathbf{m}, \tag{1}$$

where $\mathbf{W}$ is a $d \times c$ matrix, $\mathbf{x}_j$ are $c \times 1$ vectors of principal components, and $\mathbf{m}$ is a $d \times 1$ bias vector. We assume that $c \leq d \leq n$. Principal subspace methods (e.g., Cichocki and Amari, 2002;

Diamantaras and Kung, 1996) find $\mathbf{W}$, $\mathbf{x}_j$ and $\mathbf{m}$ such that the reconstruction error

$$C = \sum_{j=1}^{n} \|\mathbf{y}_j - \mathbf{W}\mathbf{x}_j - \mathbf{m}\|^2 \tag{2}$$

is minimized. In matrix notation, the data vectors and principal components can be compiled into $d \times n$ and $c \times n$ matrices $\mathbf{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \ldots\ \mathbf{y}_n]$ and $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_n]$, and y_{ij}, w_{ik} and x_{kj} denote the elements of the matrices $\mathbf{Y}$, $\mathbf{W}$ and $\mathbf{X}$, respectively. The bias matrix $\mathbf{M}$ contains n copies of the bias vector $\mathbf{m}$ as its columns. Principal subspace methods find $\mathbf{W}$ and $\mathbf{X}$ such that $\mathbf{Y} \approx \mathbf{W}\mathbf{X} + \mathbf{M}$ and the minimized cost function is the sum of the squared elements (or Frobenius norm) of matrix $\mathbf{Y} - \mathbf{W}\mathbf{X} - \mathbf{M}$:

$$C = \|\mathbf{Y} - \mathbf{W}\mathbf{X} - \mathbf{M}\|_{\mathrm{F}}^2. \tag{3}$$

Without any further constraints, there exist infinitely many ways to perform a decomposition that minimizes (2) or equivalently (3). This can be seen by noting that any rotation or scaling of $\mathbf{W}$ can be compensated by rotating or scaling $\mathbf{X}$ accordingly, leaving the product $\mathbf{W}\mathbf{X}$ the same. However, the subspace spanned by the column vectors of the matrix $\mathbf{W}$, called the *principal subspace*, is unique.

PCA finds a specific representation of the principal subspace. It is traditionally defined using the requirement that the column vectors of $\mathbf{W}$ are mutually orthogonal, have unit length and, furthermore, for each $k = 1, \ldots, c$, the first k vectors form the k-dimensional principal subspace. This makes the solution practically unique (except for changing the sign, and in the special case of having equal eigenvalues, e.g., Diamantaras and Kung, 1996; Jolliffe, 2002; Haykin, 1989). In this article, we use the term PCA for methods which seek representations (1) by minimizing the error (2). Thus we assume that once the principal subspace is found, it can be transformed into the PCA solution. This is indeed true for the case of fully observed vectors $\mathbf{y}_j$ but can be more difficult in the case with missing values, as we discuss in Section 4.

Let us now consider the same problem when the data matrix $\mathbf{Y}$ has missing entries. In this paper, we make the typical assumption that values are *missing at random* (MAR) (Little and Rubin, 1987), that is, given the observed data, the missingness does not depend on the unobserved data or latent variables. An example where the assumption does not hold is when out-of-scale measurements are marked missing. In the following example, the data matrix contains $N = 9$ observed values and 6 missing values (marked with a sign $\times$):

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & \times \\ y_{21} & y_{22} & \times & \times & y_{25} \\ \times & \times & y_{33} & y_{34} & \times \end{bmatrix}.$$

A natural extension of PCA for the case with missing values would be to find a representation such that $\mathbf{Y} \approx \mathbf{W}\mathbf{X} + \mathbf{M}$ for the observed values. The rest of the matrix $\mathbf{W}\mathbf{X} + \mathbf{M}$ can be taken as the reconstruction of missing values.

Although the PCA problem in the presence of missing values seems to be as easy as classical PCA, there are some important distinctions: 1) There is no analytical solution available since even the estimation of the data covariance matrix is nontrivial. Therefore, iterative learning procedures must be exploited. 2) The optimized cost function typically has multiple local minima and thus finding the optimal solution is more difficult. 3) There is no analytical solution even for the bias

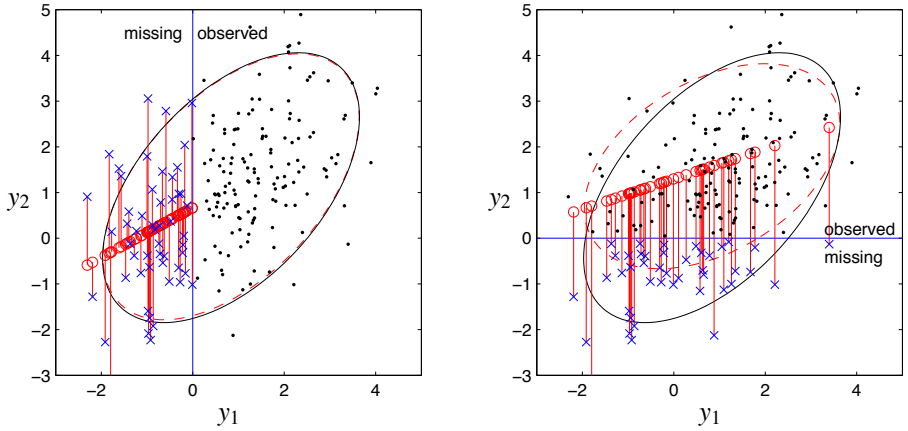


Figure 1: Two examples of PCA with missing values. In the left, the missing value mechanism is *missing at random (MAR)*, while on the right, it is *missing not at random (MNAR)*. The black dots represent fully observed samples, while the blue crosses represent observations where only the horizontal value has been observed. The red circles represent the reconstructions of the missing values produced by VBPCA. The dashed red ellipses represent the estimated data covariance. On the left, the estimated covariance is quite close to the data covariance (black ellipse).

term **m** in (1), which is not generally equal to the row-wise mean of the data matrix, as in classical PCA. 4) Standard PCA approaches can easily lead to overfitting, thus regularization is often required. 5) The algorithms may require heavy computations, especially for large-scale problems. 6) The concept of the PCA basis in the principal subspace is not easily generalized in the presence of missing values. 7) The choice of the dimensionality of the principal subspace is generally more difficult than in classical PCA. Thus, the PCA problem has many features which are usually associated with nonlinear models. This paper discusses some of these important issues providing illustrative examples and presenting ways to overcome possible difficulties.

All the methods studied in this work assume MAR, so they cannot be expected to work in the *missing not at random (MNAR)* case. Fig. 1 gives a first example of PCA with missing values. The data contain 200 samples generated from a Gaussian distribution represented by the black ellipse. In the first case (left subfigure), the vertical value is missing when the horizontal value is negative. This setting is MAR since the missingness depends only on the observed data. In the second case (right subfigure), the vertical value is missing when the vertical value itself is negative. Now the missingness depends on the missing data, and the missing value mechanism is thus MNAR. As expected, the variational Bayesian PCA (VBPCA, see Section 3.3) works well in the MAR case and gives reasonable reconstructions for the missing values. Also, the vertical part of the bias term **m** is estimated to be 1.13 which is much closer to the true value 1.10 than the row-wise mean 1.50 of the observed values in the data.

PCA in the presence of missing values can be relevant for many data sets which appear in practical applications. Sometimes, data sets contain relatively few missing observations and the

problem is to adjust standard PCA algorithms to handle partially observed instances. The choice of the algorithm is not very crucial in such a simple case, as most of the approaches would provide similar solutions. However, in other applications, the available observations can be very sparse and the modeling task is often to reconstruct the missing part from the observed data only. Examples include collaborative filtering problems which is the task of predicting preferences by using other people's preferences (e.g., Hofmann, 2004), and historical data reconstruction in climatic records (e.g., Kaplan et al., 1997).

Historically, the missing value problem in PCA was first studied by Dear (1959) who used only one component and just one imputation iteration (see below). It was based on the minimum mean-square error formulation of PCA introduced by Young (1941). Christoffersson (1970) also used a one-component model for reconstructing missing values. Wiberg (1976) first suggested to directly minimize the mean-square error of the observed part of the data. An algorithm by de Ligny et al. (1981) already worked with up to half of the values missing. The missing values problem using a multivariate normal distribution has been studied even earlier than using PCA, for instance, by Anderson (1957). More historical references can be found in the book by Jolliffe (2002).

More recently, PCA with missing values was studied by Grung and Manne (1998). They proposed using either the imputation or the faster alternating **W**–**X** algorithm (see below). They discussed the overfitting problem and suggested to delete troublesome rows or columns from data. Tipping and Bishop (1999) introduced the probabilistic formulation of PCA (PPCA). Although they mentioned shortly missing data, they did not provide the formulas to be used for incomplete data. Bishop (1999) introduced variational Bayesian PCA (VBPCA) for choosing the number of components in PCA. Raiko and Valpola (2001) reconstructed missing values with VBPCA to compare some nonlinear models to it. Oba et al. (2003) applied VBPCA for missing value estimation in gene expression profile data, and mentioned that it performs much better than the existing methods for missing value estimation.

The present article reviews possible approaches to the problem with the emphasis on probabilistic models and Bayesian methods. The rest of the article is organized as follows. In Section 2, we present classical algorithms for PCA based on minimizing the reconstruction error similar to (2) and the method based on estimation of the covariance matrix (imputation algorithm). We review how the algorithms are normally used for fully observed data and explain how they can be adapted to the case with missing values. We explain possible difficulties of the standard methods including bad locally optimal solutions, numerical problems and overfitting. Simple examples are given to illustrate these problems. In the same section, we discuss the properties of the imputation algorithm, such as implicit remedies against overfitting and overlearning and its expectation-maximization (EM) interpretation.

Section 3 presents probabilistic models for PCA which provide a good foundation for handling missing values. First, we introduce formulas for the probabilistic PCA model in the case with missing values. Then, we describe Bayesian regularization for handling problems that arise when data are sparse. We provide formulas for performing maximum a posteriori estimation and for variational Bayesian inference.

In Section 4, we propose an extension of the PCA-basis notion to the case of incomplete data. We also show how to find the PCA basis in a principal subspace estimated by different methods. Section 5 provides formulas for computing missing value reconstructions and the variance of the predicted values. We briefly discuss possible ways to select the right number of principal components to obtain reliable reconstructions.

Section 6 discusses large-scale problems and computational complexity. In case of high dimensional and very sparse data, overfitting becomes a severe problem and traditional algorithms for PCA are very slow. We introduce a novel fast optimization algorithm and show how to use it in Bayesian models.

In Section 7, different versions of PCA are compared in artificial experiments, demonstrating the effects of regularization and modeling of posterior variance. We demonstrate the importance of tricks such as fixing hyperparameters for the early learning in variational Bayesian methods. The scalability of the proposed algorithm is demonstrated by applying it to the Netflix problem. Finally, we conclude in Section 8.

1.1 Notation

The following notation is used throughout the article. Bold capital letters denote matrices, bold lower-case letters denote vectors, and normal lower-case letters denote scalars. The basic model equation is $\mathbf{y}_j \approx \mathbf{W}\mathbf{x}_j + \mathbf{m}$, where column vectors $\mathbf{y}_j$ are the data cases, $\mathbf{W}$ is the matrix that maps the principal components $\mathbf{x}_j$ to the data, and $\mathbf{m}$ is the bias vector. Indices $i = 1,\ldots,d$ and $j = 1,\ldots,n$ go over the rows and columns of $\mathbf{Y}$, respectively. The index of a principal component is denoted by $k = 1,\ldots,c$. Notation ij is used for the index of y_{ij}, the ij-th element of matrix $\mathbf{Y}$. O is the set of indices ij corresponding to *observed* values y_{ij}, O_i is the set of indices j (similarly O_j is the set of indices i) for which y_{ij} is observed. $|O_i|$ is the number of elements in O_i and $N = |O|$ is the number of observed data elements. Notation $\mathbf{A}_{i:}$ and $\mathbf{A}_{:j}$ is used for the i-th row and j-th column of a matrix $\mathbf{A}$, respectively. Both $\mathbf{A}_{i:}$ and $\mathbf{A}_{:j}$ are column vectors. The i-th row of $\mathbf{W}$ is denoted by $\mathbf{w}_i = \mathbf{W}_{i:}$, and the j-th column of $\mathbf{X}$ is $\mathbf{x}_j = \mathbf{X}_{:j}$. Table 1 lists the symbols used in the paper.

2. Least Squares Techniques

In this section, we present classical algorithms for PCA based on minimizing the reconstruction error similar to (2) and the method based on estimation of the covariance matrix.

2.1 The Cost Function

The minimum mean-square error compression of data is the formulation for PCA (Young, 1941) that can be generalized to the case with missing values in a very straightforward manner. The cost function (2) is adapted such that the sum is taken over only those indices i and j for which the data entry y_{ij} is observed (Wiberg, 1976):

$$C = \sum_{ij \in O} (y_{ij} - \hat{y}_{ij})^2 , \tag{4}$$

$$\hat{y}_{ij} = \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j + m_i = \sum_{k=1}^{c} w_{ik} x_{kj} + m_i . \tag{5}$$

In this section, we consider algorithms optimizing cost function (4), which we call the least squares (LS) approach to PCA of incomplete data.

The cost function (2) in the fully observed case has practically unique (up to an arbitrary rotation in the principal subspace) global minimum w.r.t. model parameters $\mathbf{W}, \mathbf{X}, \mathbf{m}$ provided that all eigenvalues of the sample covariance matrix are distinct. Srebro and Jaakkola (2003) showed that

d	dimensionality of the data vectors (indexed by i)
n	number of data vectors (indexed by j)
c	number of principal components (indexed by k)
$\mathbf{Y} = \{y_{ij}\}$	$d \times n$ data matrix
$\mathbf{y}_j$	d-dimensional data vector (j-th column of $\mathbf{Y}$)
$\hat{y}_{ij}$	reconstruction of the data element y_{ij}
$\mathbf{m} = \{m_i\}$	d-dimensional bias vector
$\mathbf{M}$	$= [\mathbf{m}\ \mathbf{m}\ \ldots\ \mathbf{m}]$, $d \times n$ bias matrix
$\overline{m}_i$	posterior mean of m_i (scalar)
$\widetilde{m}_i$	posterior variance of m_i (scalar)
$\mathbf{W} = \{w_{ik}\}$	$d \times c$ matrix for the mapping from principal components to the data
$\mathbf{W}_{:k}$	k-th column of matrix $\mathbf{W}$ (d-dimensional vector)
$\mathbf{w}_i$	i-th row of $\mathbf{W}$ (c-dimensional vector)
$\overline{\mathbf{w}}_i$	posterior mean of $\mathbf{w}_i$ (c-dimensional vector)
$\overline{\mathbf{W}}$	$= [\overline{\mathbf{w}}_1\ \overline{\mathbf{w}}_2\ \ldots\ \overline{\mathbf{w}}_d]^{\mathrm{T}}$, $d \times c$ matrix of posterior means of $\mathbf{w}_i$
$\mathbf{\Sigma}_{\mathbf{w}_i}$	posterior covariance of $\mathbf{w}_i$ ($c \times c$ matrix)
$\widetilde{w}_{ik}$	posterior variance of w_{ik} (scalar)
$\mathbf{X} = \{x_{kj}\}$	$c \times n$ matrix of principal components
$\mathbf{x}_j$	c-dimensional principal component vector (j-th column of $\mathbf{X}$)
$\overline{\mathbf{x}}_j$	posterior mean of $\mathbf{x}_j$ (c-dimensional vector)
$\overline{\mathbf{X}}$	$= [\overline{\mathbf{x}}_1\ \overline{\mathbf{x}}_2\ \ldots\ \overline{\mathbf{x}}_n]$, $c \times n$ matrix of posterior means of $\mathbf{x}_j$
$\mathbf{\Sigma}_{\mathbf{x}}$	posterior covariance of $\mathbf{x}_j$ (same $c \times c$ matrix for each j)
$\mathbf{\Sigma}_{\mathbf{x}_j}$	posterior covariance of $\mathbf{x}_j$ (separate $c \times c$ matrix for each j)
$\widetilde{x}_{kj}$	posterior variance of x_{kj} (scalar)
C	cost function to be minimized
O	set of indices i, j for which y_{ij} is observed
O_i	set of indices j for which y_{ij} is observed
O_j	set of indices i for which y_{ij} is observed
γ	learning rate (positive scalar)
ε_j	d-dimensional noise vector (j-th out of n vectors)
v_y	noise variance (scalar)
$\mathcal{N}(\mathbf{x}; \mu, \Sigma)$	Gaussian pdf over variable $\mathbf{x}$ with mean μ and covariance Σ
v_m	prior variance for the elements of $\mathbf{m}$ (scalar)
$v_{w,k}$	prior variance for the elements of $\mathbf{W}_{:k}$ (scalar for each k)
ξ	$= (v_y, v_{w,k}, v_m)$ hyperparameters
θ	parameters, typically $\theta = (\mathbf{W}, \mathbf{X}, \mathbf{m})$

Table 1: Symbols used in the paper. See also Section 1.1 about notation.

an extension to the weighted case, where each observation has a weight varying from 0 to 1, is a difficult non-convex problem where the cost function can have multiple local minima. We demonstrate this phenomenon for a simple data set with missing values, that is with weights either 0 (missing value) or 1 (observed value). In our example, model (1) with one principal component and fixed $\mathbf{m} = 0$ is fitted to the data

$$\mathbf{Y} = \begin{bmatrix} 0.8 & 0.8 \\ 1 & \times \\ \times & 1 \end{bmatrix} \approx \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \end{bmatrix}.$$

To discard the scaling ambiguity of the PCA solution, the column vector $\mathbf{W}$ can be restricted to have unit length. Thus, the minimized error can be represented as a function of two parameters which define a unit length vector in the three-dimensional space. The cost function in this simple example has three local minima, as shown in Fig. 2: The global minimum $\mathbf{W} = \pm[0.49\ 0.62\ 0.62]^{\mathrm{T}}$ corresponds to zero cost while the other two minima (close to $\pm[0\ 1\ 0]^{\mathrm{T}}$ and $\pm[0\ 0\ 1]^{\mathrm{T}}$) provide a non-zero error. Each of the two sub-optimal solutions reconstructs perfectly only three out of four observed values in $\mathbf{Y}$.

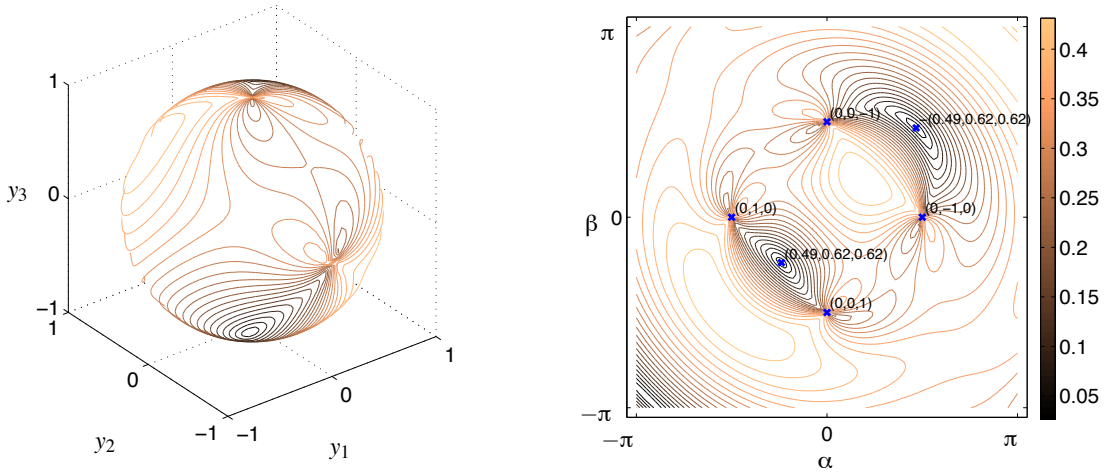


Figure 2: Example of local minima for the cost function (4). Data are three-dimensional ($d = 3$), the model has one principal component ($c = 1$) and therefore the PCA solution can be defined by a unit length vector $\mathbf{W}$. Left: The cost function plotted on a surface of unit length vectors. Right: The same plot using the Euler vector representation: The matrix $\mathbf{W}$ is constructed from α and β as $\mathbf{W} = e^{\mathbf{A}}[1\ 0\ 0]^{\mathrm{T}}$ with $\mathbf{A}$ a 3×3 matrix with four nonzero elements $a_{12} = -a_{21} = \alpha$ and $a_{13} = -a_{31} = \beta$. Coordinates in brackets correspond to coordinates in Euclidian space in the left hand side plot.

The cost function (4) can be minimized using any optimization procedure. Two possible approaches are presented in the following.

2.2 Alternating W–X Algorithm

Complete data. It is possible to optimize the cost function (3), and therefore to find the principal subspace, by updating $\mathbf{W}$ and $\mathbf{X}$ alternately. When either of these matrices is fixed, the other one

can be obtained from an ordinary least squares problem. We will further refer to this approach as the *alternating algorithm*.

The algorithm alternates between the updates

$$\mathbf{X} = (\mathbf{W}^{\mathrm{T}}\mathbf{W})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{Y}, \tag{6}$$

$$\mathbf{W} = \mathbf{Y}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}. \tag{7}$$

This iteration can be efficient when only a few principal components are needed, that is $c \ll d$ (Roweis, 1998). The bias vector $\mathbf{m}$ is the row-wise mean of the data and the above equations assume that it has been subtracted from the data as a preprocessing step.

Incomplete data. Grung and Manne (1998) studied the alternating algorithm in the case of missing values. In order to get the accurate least squares solution, we include the bias term into the estimation procedure, yielding the update rules

$$\mathbf{x}_j = \Big(\sum_{i\in O_j} \mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}\Big)^{-1} \sum_{i\in O_j} \mathbf{w}_i(y_{ij} - m_i)\,, \quad j = 1,\ldots,n, \tag{8}$$

$$m_i = \frac{1}{|O_i|}\sum_{j\in O_i}\left[y_{ij} - \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j\right], \tag{9}$$

$$\mathbf{w}_i = \Big(\sum_{j\in O_i} \mathbf{x}_j\mathbf{x}_j^{\mathrm{T}}\Big)^{-1} \sum_{j\in O_i} \mathbf{x}_j(y_{ij} - m_i)\,, \quad i = 1,\ldots,d\,, \tag{10}$$

where m_i is the i-th element of $\mathbf{m}$.

2.3 Gradient Descent Algorithm

Complete data. The basis of neural network implementation of PCA learning rules is a gradient descent optimization procedure. Such algorithms work online, processing only one input vector $\mathbf{y}_j$ at once. The learning rules implement *stochastic* gradient descent and the algorithms will eventually converge to a basis in the principal subspace. Each data vector may have to be processed several times for convergence. The same can also be implemented in a batch procedure.

Using gradient descent for minimization of (4) w.r.t. $\mathbf{W}$ yields the update rule

$$\mathbf{W} \leftarrow \mathbf{W} + \gamma(\mathbf{Y} - \mathbf{W}\mathbf{X})\mathbf{X}^{\mathrm{T}}\,, \tag{11}$$

where $\gamma > 0$ is called the learning rate. Minimization w.r.t. matrix $\mathbf{X}$ can be performed using the least squares solution in (6). Some neural algorithms use learning rules which either explicitly orthogonalize $\mathbf{W}$ or which yield an orthogonal $\mathbf{W}$ at the convergence. Then the update of $\mathbf{X}$ can be simplified to $\mathbf{X} = \mathbf{W}^{\mathrm{T}}\mathbf{Y}$, which together with (11) is the batch version of the Oja learning algorithm (Oja, 1983; Diamantaras and Kung, 1996). The bias $\mathbf{m}$ is again removed from the data as a preprocessing step.

Incomplete data. In the presence of missing values, the gradient descent update rule for $\mathbf{W}$ is

$$\mathbf{W} \leftarrow \mathbf{W} - \gamma\frac{\partial C}{\partial \mathbf{W}}\,, \qquad \text{with} \qquad \frac{\partial C}{\partial w_{ik}} = -2\sum_{j\in O_i}(y_{ij} - \hat{y}_{ij})x_{kj}\,,$$

where $\hat{y}_{ij}$ is given in (5). Matrix $\mathbf{X}$ could be updated using (8) but a gradient-based update can also be used:

$$\mathbf{X} \leftarrow \mathbf{X} - \gamma \frac{\partial C}{\partial \mathbf{X}}, \qquad \text{with} \qquad \frac{\partial C}{\partial x_{kj}} = -2 \sum_{i \in O_j} (y_{ij} - \hat{y}_{ij}) w_{ik}.$$

The bias term $\mathbf{m}$ can be updated using (9). As we discuss in Section 6.1, gradient-based learning can be computationally more efficient than the alternating algorithm because there is no need to compute matrices $\left(\sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}}\right)^{-1}$ and $\left(\sum_{j \in O_i} \mathbf{x}_j \mathbf{x}_j^{\mathrm{T}}\right)^{-1}$ in (8) and (10).

2.4 Numerical Problems and Overfitting

Least squares PCA methods can work well for data sets with few missing values but they may not be applicable to sparse data sets because of severe problems with overfitting. Suppose that for some j the number $|O_j|$ of observed measurements y_{ij} is smaller than the number of principal components c. Then, the corresponding least square problem is ill posed: the matrix $\sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}}$ is rank deficient and Equation (8) cannot be used. Moreover, even if $|O_j|$ is greater than c, matrix $\sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}}$ may be badly conditioned and the corresponding $\mathbf{x}_j$ can become infinitely large. This means that the parameters would be overfitted to explain well a few observed values but the generalization ability of the model would be poor (e.g., reconstruction of missing data would be very inaccurate). This is exactly what happens in the vicinity of the two local minima in the example in Fig. 2.

The same problem can happen when the rows of $\mathbf{W}$ are estimated using, for example, (10): matrix $\sum_{j \in O_i} \mathbf{x}_j \mathbf{x}_j^{\mathrm{T}}$ can be rank deficient or badly conditioned. This is more probable when some rows of $\mathbf{Y}$ contain very few measurements. However, these problems can generally appear for any data set and for any algorithm optimizing the cost function (4).

Overfitting problems can be avoided by using proper regularization. A common way to prevent unbounded growth of model parameters is to add terms which penalize large parameter values into the cost function. The amount of regularization can be determined, for instance, by cross-validation. Another possibility is to use probabilistic methods (see Section 3) in which the extra penalty terms come naturally from a probabilistic model.

2.5 Method Using Singular Value Decomposition

Complete data. Perhaps the most popular approach to PCA is based on singular value decomposition (SVD) of the data matrix or (equivalently) eigen-decomposition of the sample covariance matrix. SVD of the data matrix is given by:

$$\mathbf{Y} = \mathbf{U} \Sigma \mathbf{V}^{\mathrm{T}},$$

where $\mathbf{U}$ is a $d \times d$ orthogonal matrix, $\mathbf{V}$ is an $n \times n$ orthogonal matrix and Σ is a $d \times n$ pseudo-diagonal matrix (diagonal if $d = n$) with the singular values on the main diagonal (e.g., Haykin, 1989). The PCA solution is obtained by selecting the c largest singular values from Σ, by forming $\mathbf{W}$ from the corresponding c columns of $\mathbf{U}$, and $\mathbf{X}$ from the corresponding c rows of $\Sigma \mathbf{V}^{\mathrm{T}}$ (Jolliffe, 2002). The bias $\mathbf{m}$ is again removed from the data as a preprocessing step.

PCA can equivalently be defined using the eigen-decomposition of the $d \times d$ covariance matrix $\mathbf{C}$ of the column vectors of the data matrix $\mathbf{Y}$:

$$\mathbf{C} = \frac{1}{n} \mathbf{Y} \mathbf{Y}^{\mathrm{T}} = \mathbf{U} \mathbf{D} \mathbf{U}^{\mathrm{T}}.$$

The diagonal matrix $\mathbf{D}$ contains the eigenvalues of $\mathbf{C}$, and the columns of the matrix $\mathbf{U}$ contain the unit-length eigenvectors of $\mathbf{C}$ in the same order (Cichocki and Amari, 2002; Diamantaras and Kung, 1996; Jolliffe, 2002; Haykin, 1989). Again, the columns of $\mathbf{U}$ corresponding to the largest eigenvalues are taken as $\mathbf{W}$, and $\mathbf{X}$ is computed as $\mathbf{W}^{\mathrm{T}}\mathbf{Y}$. This approach can be more efficient for cases where $d \ll n$, since it avoids the computation of the $n \times n$ matrix $\mathbf{V}$.

Incomplete data. The same approach cannot be directly used in the presence of missing values. Estimating the covariance matrix $\mathbf{C}$ becomes difficult. Let us consider a simple (but incorrect) estimate where we just leave out terms with missing values from the average for each element of $\mathbf{C}$. For the data matrix below, we get

$$\mathbf{Y} = \begin{bmatrix} -1 & +1 & 0 & 0 & \times \\ -1 & +1 & \times & \times & 0 \\ \times & \times & -1 & +1 & \times \end{bmatrix}, \quad \mathbf{C} = \frac{1}{n}\mathbf{Y}\mathbf{Y}^{\mathrm{T}} = \begin{bmatrix} 0.5 & 1 & 0 \\ 1 & 0.667 & \times \\ 0 & \times & 1 \end{bmatrix}.$$

There are at least two problems. First, the estimated covariance 1 between the first and second components is larger than their estimated variances 0.5 and 0.667. This clearly leads to the situation where the covariance matrix is not positive (semi)definite and some of its eigenvalues are negative. Secondly, the covariance between the second and the third component could not be estimated at all. For these reasons, this is a viable option only if the number of missing values is not significant. There are many algorithms for finding a proper estimate of the covariance matrix (e.g., Ghahramani and Jordan, 1994; Boscardin and Zhang, 2004) but they are computationally intensive iterative algorithms.

A simple alternative is an iterative procedure which performs PCA more directly. It alternates between imputing the missing values in $\mathbf{Y}$ and applying standard PCA to the infilled (complete) data matrix $\mathbf{Y}_c$ (e.g., Jolliffe, 2002). Initially, the missing values can be replaced, for example, by the row-wise means of $\mathbf{Y}$. The covariance matrix of the complete data $\mathbf{Y}_c$ can be estimated without the problems mentioned above, and $\mathbf{W}$ can be computed using its eigen-decomposition. The bias term $\mathbf{m}$ can be updated as the row-wise mean of $\mathbf{Y}_c$. Next, the principal components $\mathbf{X}$ are calculated:

$$\mathbf{X} = \mathbf{W}^{\mathrm{T}}(\mathbf{Y}_c - \mathbf{M}),$$

the reconstructions can be used as a better estimate for the missing values:

$$\mathbf{Y}_c = \begin{cases} \mathbf{Y} & \text{for observed values} \\ \mathbf{W}\mathbf{X} + \mathbf{M} & \text{for missing values} \end{cases}$$

and PCA can be applied again to the updated data matrix $\mathbf{Y}_c$. This process can be iterated until convergence. We will further refer to this approach as the *imputation algorithm*. In Appendix A, we show that the imputation algorithm also minimizes the cost function (4) and that it implements the EM steps for a simple probabilistic model. This approach requires the use of the complete data matrix, and therefore it is computationally very expensive for large-scale problems.

Although the imputation algorithm belongs to the class of least-squares PCA algorithms (i.e., it does not use explicit regularization), it still provides some remedy against overfitting. Performing SVD of a complete data matrix is equivalent to minimizing a cost function which is a sum of two terms: the reconstruction error for the observed values (which is the true minimized error) and the reconstruction error for the infilled missing data (which is a penalty term forcing the new reconstructions of missing data to be close to the infilled values). Initialization of the reconstructions

with the row-wise means of the data matrix introduces a bias in favor of the most "conservative" solutions. In practice, there can be infinitely many solutions that provide the same reconstruction error for the observed part but that have different reconstructions for the missing part. In such cases, the effect of the initialization can last even if the algorithm is iterated indefinitely.

Our experiments show that the imputation algorithm also has resistance against overlearning. By overlearning we mean situations when the training error decreases but the generalization ability of the model gets worse. Badly overfitted solutions correspond to regions in the parameter space where a small decrease of the training error can cause a large increase of the test error. However, the penalty term (which keeps the reconstructions of missing data close to the infilled values) makes the steps shorter and learning can practically stop once the algorithm enters regions where the training error changes very little. Thus, this regularization effect is due to early stopping.

3. Probabilistic Models for PCA

In this section, we presents probabilistic models for PCA which provide a good foundation for handling missing values.

3.1 Probabilistic PCA

The probabilistic formulation of PCA offers a number of benefits, including well-founded regularization, model comparison, interpretation of results, and extendability. *Probabilistic PCA* (Tipping and Bishop, 1999) explicitly includes the noise term in a generative model

$$\mathbf{y}_j = \mathbf{W}\mathbf{x}_j + \mathbf{m} + \varepsilon_j \,. \tag{12}$$

Both the principal components $\mathbf{x}_j$ and the noise ε_j are assumed normally distributed:

$$p(\mathbf{x}_j) = \mathcal{N}(\mathbf{x}_j; 0, \mathbf{I}) \,, \tag{13}$$

$$p(\varepsilon_j) = \mathcal{N}(\varepsilon_j; 0, v_y\mathbf{I}) \,, \tag{14}$$

where $\mathcal{N}(\mathbf{x}; \mu, \Sigma)$ denotes the normal probability density function (pdf) over variable $\mathbf{x}$ with mean μ and covariance Σ. The parameters of the model include $\mathbf{W}$, $\mathbf{m}$ and v_y. The model can be identified by finding the maximum likelihood (ML) estimate for the model parameters using the EM algorithm (Dempster et al., 1977).

Complete data. The E-step estimates the conditional distribution of the hidden variables $\mathbf{X}$ given the data and the current values of the model parameters:

$$p(\mathbf{X}|\mathbf{Y}, \mathbf{W}, v_y) = \prod_{j=1}^{n} \mathcal{N}\left(\mathbf{x}_j; \overline{\mathbf{x}}_j, \Sigma_{\mathbf{x}_j}\right) \,,$$

where the covariance matrix $\Sigma_{\mathbf{x}_j}$ is same for all $\mathbf{x}_j$:

$$\Sigma_{\mathbf{x}_j} = \Sigma_{\mathbf{x}} = v_y\left(v_y\mathbf{I} + \mathbf{W}^{\mathrm{T}}\mathbf{W}\right)^{-1}, \quad j = 1, \ldots n \,,$$

and the means $\overline{\mathbf{x}}_j$ can be summarized in the columns of matrix

$$\overline{\mathbf{X}} = \frac{1}{v_y}\Sigma_{\mathbf{x}}\mathbf{W}^{\mathrm{T}}\mathbf{Y} \,. \tag{15}$$

The M-step re-estimates the model parameters as

$$\mathbf{W} = \mathbf{Y}\overline{\mathbf{X}}^{\mathrm{T}}(\overline{\mathbf{X}\mathbf{X}}^{\mathrm{T}} + n\Sigma_{\mathbf{x}})^{-1}, \tag{16}$$
$$v_y = \frac{1}{nd}\sum_{i=1}^{d}\sum_{j=1}^{n}\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}}\overline{\mathbf{x}}_j\right)^2 + \frac{1}{d}\,\mathrm{tr}(\mathbf{W}\Sigma_{\mathbf{x}}\mathbf{W}^{\mathrm{T}})\,.$$

Note that v_y is updated as the mean-square error plus the extra term which accounts for the uncertainty in $\mathbf{X}$.

Tipping and Bishop (1999) showed that $\mathbf{W}$ converges to the principal subspace. In the zero-noise limit, that is for $v_y \to 0$, the iterations (15)–(16) reduce to the least squares projections in (6)–(7) (Roweis, 1998). Note also that the ML estimator for $\mathbf{m}$ is given by the row-wise mean of the data (Tipping and Bishop, 1999). Therefore, it can be computed once and removed from the data in the preprocessing step.

Incomplete data. The generalization to the case with missing values happens as follows. First we write an element-wise version of the probabilistic model in (12)–(14):

$$\begin{aligned} y_{ij} &= \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j + m_i + \varepsilon_{ij}\,, \qquad \forall ij \in O,\\ p(\mathbf{x}_j) &= \mathcal{N}(\mathbf{x}_j; 0, \mathbf{I})\,,\\ p(\varepsilon_{ij}) &= \mathcal{N}(\varepsilon_{ij}; 0, v_y)\,. \end{aligned}$$

We treat $\mathbf{w}_i$, m_i, and v_y as model parameters and $\mathbf{x}_j$ as latent variables and apply the standard EM algorithm to arrive at the following update rules:

$$\Sigma_{\mathbf{x}_j} = v_y\Big(v_y\mathbf{I} + \sum_{i\in O_j}\mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}\Big)^{-1}, \tag{17}$$
$$\overline{\mathbf{x}}_j = \frac{1}{v_y}\Sigma_{\mathbf{x}_j}\sum_{i\in O_j}\mathbf{w}_i(y_{ij} - m_i)\,, \qquad j = 1,\ldots,n,$$
$$m_i = \frac{1}{|O_i|}\sum_{j\in O_i}\left[y_{ij} - \mathbf{w}_i^{\mathrm{T}}\overline{\mathbf{x}}_j\right],$$
$$\mathbf{w}_i = \Big(\sum_{j\in O_i}\left[\overline{\mathbf{x}}_j\overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j}\right]\Big)^{-1}\sum_{j\in O_i}\overline{\mathbf{x}}_j(y_{ij} - m_i)\,, \qquad i = 1,\ldots,d, \tag{18}$$
$$v_y = \frac{1}{N}\sum_{ij\in O}\left[\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}}\overline{\mathbf{x}}_j - m_i\right)^2 + \mathbf{w}_i^{\mathrm{T}}\Sigma_{\mathbf{x}_j}\mathbf{w}_i\right].$$

There are several distinctions compared to the fully observed data: 1) The optimal $\mathbf{m}$ depends on other parameters and therefore it has to be updated in the iterative procedure. 2) The covariance $\Sigma_{\mathbf{x}_j}$ is different for each $\mathbf{x}_j$. 3) Each row of $\mathbf{W}$ is recomputed based only on those columns of $\overline{\mathbf{X}}$ which contribute to the reconstruction of the *observed* values in the corresponding row of the data matrix. The same applies to the computation of the columns of $\overline{\mathbf{X}}$. Thus, the computations required for incomplete data are generally heavier.

3.2 Examples of Overfitting with Probabilistic PCA

PPCA provides some remedy against overfitting, in contrast to the least squares approach. First, a nonzero noise level v_y regularizes badly conditioned matrices $\sum_{i\in O_j}\mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}$ in (17). Second, even

if the noise level is small, badly conditioned matrices $\sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}}$ result in large values both in the means $\overline{\mathbf{x}}_j$ and in the covariance matrices $\mathbf{\Sigma}_{\mathbf{x}_j}$. This diminishes the effect of large values in $\overline{\mathbf{x}}_j$ when $\mathbf{W}$ is re-estimated using (18).

However, PPCA can overfit, for example, if the estimated number of principal components is unreasonably large. In the example in Fig. 3, PPCA with one ($c = 1$) principal component is applied to two-dimensional ($d = 2$) data and each data vector is only partly observed (i.e., either y_{1j} or y_{2j} is known for all js). The observations are represented by triangles placed on the two axes in Fig. 3. The data points reconstructed with a trained PPCA model lie on a line, thus the model invents correlations that are not observed in the data. In fact, there are infinitely many PPCA models which are equally good and the final solution (and hence the reconstructions) depends on initialization. The situation would not change significantly if there were a few fully observed data vectors. Then, the trained model would be defined by those few samples and it would not generalize well for new data.

A similar overfitting example is presented in Fig. 4. There, three-dimensional data are described well by a model with $c = 1$ principal component. The first two rows of $\mathbf{Y}$ are fully observed while there are only two measurements in the third row (Fig. 4a). The reconstructions of the third row of $\mathbf{Y}$ are very inaccurate because the model relies only on the two available observations.

3.3 Variational Bayesian PCA (VBPCA)

A common way to cope with the overfitting problem is to penalize parameter values which correspond to more complex explanations of the data. A natural regularization in PCA is using penalization of large values in matrices $\mathbf{W}$ and $\mathbf{X}$. In the Bayesian formulation, this is equivalent to introducing a prior over the model parameters. For example, the PPCA model in (12)–(14) can be complemented with Gaussian priors over the elements of vector $\mathbf{m}$ and matrix $\mathbf{W}$:

$$p(\mathbf{m}) = \mathcal{N}(\mathbf{m}; 0, v_m \mathbf{I}), \tag{19}$$

$$p(\mathbf{W}) = \prod_{k=1}^{c} \mathcal{N}(\mathbf{W}_{:k}; 0, v_{w,k} \mathbf{I}). \tag{20}$$

Here, we use a zero mean prior for $\mathbf{m}$ for the sake of simplicity. Including a mean hyperparameter μ, that is $p(\mathbf{m}) = \prod_i \mathcal{N}(m_i; \mu, v_m)$, can be useful in practice.

The model (20) uses a shared prior for all elements in the same column of $\mathbf{W}$, parameterized with $v_{w,k}$. This is done to allow automatic selection of the right number of components needed for PCA. The hyperparameters v_m, $v_{w,k}$ can also be updated during learning (e.g., using the evidence framework or variational approximations). If the evidence of the relevance of the k-th principal component for reliable data modeling is weak, the corresponding $v_{w,k}$ should tend to zero. This is called *automatic relevance determination* (e.g., Bishop, 2006). The prior in (19)–(20) also introduces a bias towards the PCA basis within the principal subspace (Luttinen and Ilin, 2010).

Let us assume that we perform ML estimation of the hyperparameters $\xi = (v_y, v_{w,k}, v_m)$ in the probabilistic model defined in (12)–(14) and (19)–(20). This can be done using the EM-algorithm if one treats the model parameters $\theta = (\mathbf{W}, \mathbf{X}, \mathbf{m})$ as hidden random variables. Implementation of the EM algorithm would require the computation of the posterior of the hidden variables $p(\theta|\mathbf{Y}, \xi)$ on the E-step. Unfortunately, the true posterior $p(\theta|\mathbf{Y}, \xi)$ does not have an analytic form and one possible solution is to approximate it with a simpler pdf $q(\theta)$. This is justified by the variational view of the EM algorithm (Neal and Hinton, 1999).

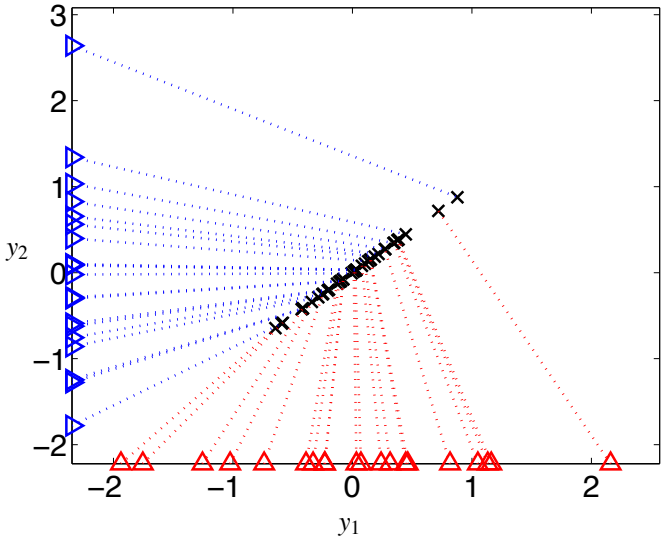


Figure 3: A simple example of overfitting with PPCA: data are two-dimensional and each sample contains only one observed value. The measurements are marked with blue and red triangles at the two axes. The reconstructions provided by a trained PPCA model are shown with crosses.

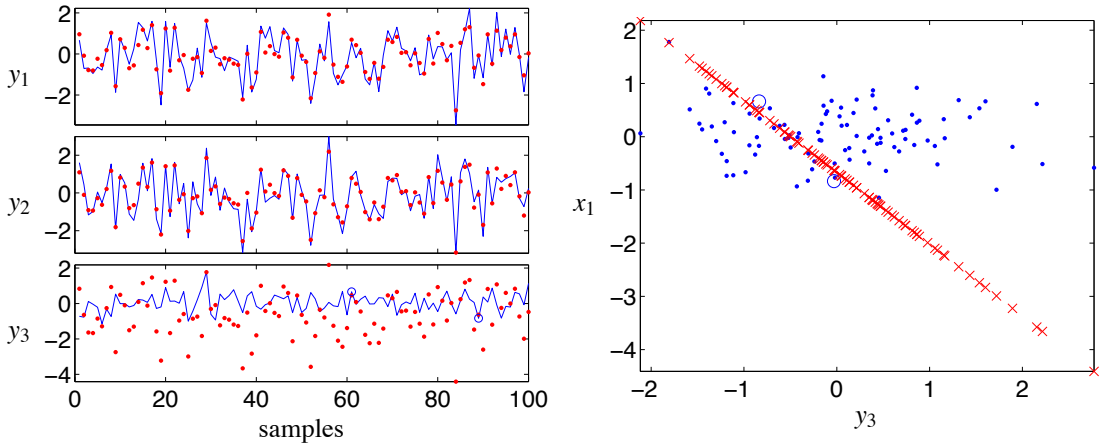


Figure 4: Example of inaccurate reconstruction with PPCA. Left: The "true" values in the three rows of $\mathbf{Y}$ are shown with blue lines. The first two rows are fully observed, the only two observations in the third row are marked with circles. The PPCA reconstruction is shown with red dots. Right: The blue dots is the scatter plot of the third row of $\mathbf{Y}$ (x-axis) against the principal component estimated by PPCA (y-axis). The dots corresponding to the two observations are marked with circles. The red crosses is the reconstruction of the third row (x-axis) against the estimated principal component (y-axis). The estimated correlation are due to the two fully observed data vectors.

Using the variational approach (Wallace, 1990; Hinton and van Camp, 1993), the E-step is modified to update the approximation $q(\theta)$ so as to minimize the cost function

$$C(q(\theta),\xi) = \int q(\theta)\log\frac{q(\theta)}{p(\mathbf{Y},\theta|\xi)}d\theta = \int q(\theta)\log\frac{q(\theta)}{p(\theta|\mathbf{Y},\xi)}d\theta - \log p(\mathbf{Y}|\xi)\,. \tag{21}$$

On the M-step, the approximation $q(\theta)$ is used as it was the actual posterior $p(\theta|\mathbf{Y},\xi)$ in order to increase likelihood $p(\mathbf{Y}|\xi)$. This can be seen as minimization of (21) w.r.t. ξ (Neal and Hinton, 1999).

The first term in (21) is the Kullback-Leibler divergence between the true posterior and its approximation. Since it is always non-negative, the cost function provides a lower bound of the log-likelihood:

$$\log p(\mathbf{Y}|\xi) \geq -C(q(\theta),\xi)\,. \tag{22}$$

This property can be used to compare solutions corresponding to different local minima of (21). More details on variational methods can be found, for example, in the book by Bishop (2006).

The variational Bayesian (VB) approach to PCA can be implemented by minimization of (21) w.r.t. the approximating pdf $q(\theta)$ and ξ.[1] The complexity of the cost function (21) depends on the form of the approximating pdf $q(\theta)$. A computationally convenient form for the PCA model is

$$q(\theta) = \prod_{i=1}^{d} q(m_i)\prod_{i=1}^{d} q(\mathbf{w}_i)\prod_{j=1}^{n} q(\mathbf{x}_j)\,. \tag{23}$$

Then, the cost function can be minimized alternately w.r.t. one factor $q(\theta_i)$ in (23) while keeping the other ones fixed. Because we use conjugate priors in (19)–(20), it is possible to find *optimal* pdfs $q(\theta_i)$ on each step: The optimal $q(m_i)$, $q(\mathbf{w}_i)$ and $q(\mathbf{x}_j)$ are Gaussian and their update boils down to re-estimation of the corresponding means and covariance matrices.

In Appendix C, we present the update rules for the resulting algorithm, which we call VBPCA. There, we assume incomplete data sets, the update rules for the fully observed data can be found in the paper by Bishop (1999). Note the resemblance of the learning rules to the EM algorithm applied to PPCA. The mean parameters $\overline{\mathbf{W}}$, $\overline{\mathbf{X}}$, $\overline{\mathbf{m}}$ of the approximating pdfs can be used as estimates of the corresponding model parameters. The covariance matrices $\Sigma_{\mathbf{w}_i}$, $\Sigma_{\mathbf{x}_j}$ and variances $\widetilde{m}_i$ reflect the uncertainty about the corresponding quantities.

The advantages of the VB approach can be summarized in the following.

- VB learning is sensitive to posterior probability mass rather than posterior probability density, which makes it more resistant against overfitting compared to point estimation (e.g., PPCA or MAPPCA presented in Section 6.4). Typically, overfitted solutions correspond to high peaks in the posterior density while robust solutions contain much of the posterior probability mass in their neighborhood.

- The method provides information about uncertainty for the unknown quantities (in the estimated posterior covariance matrices). This can be useful to detect unreliable results similar to the one presented in Fig. 4. For example, the uncertainty of the reconstructions of missing values can be estimated as we show in Section 5.

1. In the fully Bayesian treatment, the hyperparameters ξ are also assigned priors and they are treated equally to the rest of the parameters θ. We omit the effect of the prior for the hyperparameters to simplify the equations.

- The VB cost function can be used for comparison between different solutions based on (22): A smaller cost yields a greater lower bound of the solution evidence. This is a useful feature for PCA of incomplete data as, in this problem, even simpler models can provide cost functions with multiple local minima. A greater lower bound does not guarantee a better solution, but still it seems to be a reliable measure in practice, see Figure 11 for an example.

4. Rotation in the Principal Subspace to the PCA Basis

A useful property of classical PCA is that the principal components are ordered by the amount of data variance they explain, which allows an intuitive interpretation of the results. Such a representation is quite trivial for complete data sets but the extensions to the case of incomplete data is not straightforward. In this section, we show how to find such a basis in the principal subspace estimated by different algorithms for incomplete data. We refer to this basis as the PCA basis in the principal subspace.

One can define the PCA basis for *complete data* as the solution which minimizes (2) and which satisfies the following conditions: 1) the principal components (in the rows of $\mathbf{X}$) are zero mean, mutually uncorrelated and scaled to unit variance:

$$\frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j = 0, \tag{24}$$

$$\frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j\mathbf{x}_j^{\mathrm{T}} = \frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}} = \mathbf{I} \tag{25}$$

and 2) matrix $\mathbf{W}$ has mutually orthogonal columns which are ordered according to their norms:

$$\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathrm{diag}(\mathbf{s}), \tag{26}$$

$$s_k \geq s_l\,, \quad k < l\,, \tag{27}$$

where $\mathrm{diag}(\mathbf{s})$ denotes a diagonal matrix with diagonal elements $s_k = \|\mathbf{W}_{:k}\|^2$. Note that the normalized columns of $\mathbf{W}$ and their squared norms s_k are the eigenvectors and eigenvalues of the data covariance matrix.

We propose to use the same conditions (24)–(27) to define the PCA basis in the principal subspace estimated for *incomplete data*. Note the important difference to the complete data case: The PCA basis for complete data is unique (assuming distinct eigenvalues of the data covariance matrix) and it does not depend on the dimensionality for the assumed principal subspace. This cannot be guaranteed for incomplete data case: The principal subspace estimated with the same algorithm using fewer components may differ from the leading directions of the PCA basis found in the subspace with more components.

In the following, we show how to transform a solution found with different algorithms to the PCA basis such that conditions (24)–(27) are satisfied.

4.1 Least Squares Approaches

The first step to transform a solution $\{\mathbf{W},\mathbf{m},\mathbf{X}\}$ found by a least squares algorithm (as presented in Sections 2.2-2.3) is to center the rows in $\mathbf{X}$ to zero mean:

$$\mathbf{x}_j \leftarrow \mathbf{x}_j - \mu, \tag{28}$$

$$\mathbf{m}_{\text{pca}} = \mathbf{m} + \mathbf{W}\mu, \tag{29}$$

with $\mu = \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j$. This ensures (24). The next step is to transform linearly $\mathbf{W}$ and $\mathbf{X}$:

$$\mathbf{W}_{\text{pca}} = \mathbf{W}\mathbf{U}\mathbf{D}_x^{1/2}\mathbf{V}, \tag{30}$$

$$\mathbf{X}_{\text{pca}} = \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{X}, \tag{31}$$

where matrices $\mathbf{U}, \mathbf{D}_x$ are computed by eigen-decomposition

$$\frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}} = \mathbf{U}\mathbf{D}_x\mathbf{U}^{\mathrm{T}} \tag{32}$$

and $\mathbf{V}$ is calculated by eigen-decomposition

$$\mathbf{D}_x^{1/2}\mathbf{U}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{U}\mathbf{D}_x^{1/2} = \mathbf{V}\mathbf{D}_w\mathbf{V}^{\mathrm{T}}. \tag{33}$$

It is easy to show that the transformed solution satisfies (25)–(26):

$$\begin{aligned}
\frac{1}{n}\mathbf{X}_{\text{pca}}\mathbf{X}_{\text{pca}}^{\mathrm{T}} &= \frac{1}{n}\mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{U}\mathbf{D}_x^{-1/2}\mathbf{V} \\
&= \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{D}_x\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{D}_x^{-1/2}\mathbf{V} = \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{D}_x\mathbf{D}_x^{-1/2}\mathbf{V} = \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}, \\
\mathbf{W}_{\text{pca}}^{\mathrm{T}}\mathbf{W}_{\text{pca}} &= \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{1/2}\mathbf{U}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{U}\mathbf{D}_x^{1/2}\mathbf{V} = \mathbf{V}^{\mathrm{T}}\mathbf{V}\mathbf{D}_w\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{D}_w.
\end{aligned}$$

4.2 Probabilistic PCA

We show in Appendix B that the following conditions hold at the convergence of PPCA:

$$\frac{1}{n}\sum_{j=1}^{n}\overline{\mathbf{x}}_j = 0, \tag{34}$$

$$\Sigma_* = \frac{1}{n}\sum_{j=1}^{n}\left[\overline{\mathbf{x}}_j\overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j}\right] = \mathbf{I}, \tag{35}$$

which can be seen as the analogue of (24)–(25). Note that Σ_* becomes the sample covariance matrix of the principal components in the noiseless limit, when $v_y \to 0$. The extra requirements (26)–(27) can be used to define the (practically) unique PCA basis for a PPCA solution. They resolve the rotational ambiguity caused by the fact that any orthogonal transformation of the principal subspace does not change Σ_* in (35).

One can easily perform a transformation of a PPCA solution such that (34)–(35) and (26)–(27) hold. This is done by first normalizing $\overline{\mathbf{x}}_j$ to zero mean and updating the bias term similarly to (28)–(29). Then, a rotation of the subspace is performed similarly to (30)–(33) with the exception that

$\frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}}$ is replaced with Σ_*. Finally, one can rotate the posterior covariance matrices of the estimated principal components:

$$\Sigma_{\mathbf{x}_j,\mathrm{pca}} = \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{U}^{\mathrm{T}}\Sigma_{\mathbf{x}_j}\mathbf{U}\mathbf{D}_x^{-1/2}\mathbf{V}\,, \tag{36}$$

which follows from (31).

Although the optimal PPCA solution satisfy at least conditions (34)–(35), convergence of the learning algorithm to the optimal solution can be very slow. Therefore, performing the described transformations during learning can speed up learning. In the experiments, we use the transformations after each iteration.

4.3 VBPCA

It can be shown (Luttinen and Ilin, 2010) that VBPCA described in Section 3.3 converges to the solution which satisfies (34)–(35) and the condition analogous to (26):

$$\sum_{i=1}^{d}\left[\overline{\mathbf{w}_i}\overline{\mathbf{w}_i}^{\mathrm{T}} + \Sigma_{\mathbf{w}_i}\right] = \mathrm{diag}(\mathbf{s})\,, \tag{37}$$

where $\overline{\mathbf{w}_i}$, $\Sigma_{\mathbf{w}_i}$ are the posterior means and covariance matrices for the rows of matrix $\mathbf{W}$ (see Appendix C for notation details). Thus, one can simply use the ordering requirement (27) to define a meaningful basis for a VBPCA solution. Performing a transformation that ensures (34)–(35) and (37) during learning can speed up convergence (Luttinen and Ilin, 2010). The transformation can be performed similarly to (30)–(33) with the exception that $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ is replaced with $\sum_{j=1}^{n}\left[\overline{\mathbf{x}}_j\overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j}\right]$ in (32) and $\mathbf{W}^{\mathrm{T}}\mathbf{W}$ is replaced with $\sum_{i=1}^{d}\left[\overline{\mathbf{w}_i}\overline{\mathbf{w}_i}^{\mathrm{T}} + \Sigma_{\mathbf{w}_i}\right]$ in (33). Finally, one can rotate the posterior covariance matrices using (36) and

$$\Sigma_{\mathbf{w}_i,\mathrm{pca}} = \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{1/2}\mathbf{U}^{\mathrm{T}}\Sigma_{\mathbf{w}_i}\mathbf{U}\mathbf{D}_x^{1/2}\mathbf{V}\,.$$

In the experiments, we use the transformations after each iteration.

5. Reconstruction of Missing Values and Selection of Model Rank

PCA is a popular approach to the problem of missing value reconstruction. The estimated principal components capture the correlations between different variables, which allows for reconstruction of missing values from the observed ones. The models discussed in this article compute reconstructions using (5), in which $\mathbf{w}_i, m_i, \mathbf{x}_j$ are replaced with the respective posterior means $\overline{\mathbf{w}}_i, \overline{m}_i, \overline{\mathbf{x}}_j$ when applicable. Additionally, one can estimate the uncertainty of the predictions for PPCA and VBPCA by computing its variance

$$\widetilde{y}_{ij} = \widetilde{m}_i + \overline{\mathbf{w}}_i^{\mathrm{T}}\Sigma_{\mathbf{x}_j}\overline{\mathbf{w}}_i + \overline{\mathbf{x}}_j^{\mathrm{T}}\Sigma_{\mathbf{w}_i}\overline{\mathbf{x}}_j + \mathrm{tr}(\Sigma_{\mathbf{x}_j}\Sigma_{\mathbf{w}_i})\,,$$

where $\widetilde{m}_i$, $\Sigma_{\mathbf{x}_j}$, $\Sigma_{\mathbf{w}_i}$ represent the posterior uncertainties of the respective parameters and $\Sigma_{\mathbf{w}_i}$ is zero for PPCA. This is a useful feature of the probabilistic methods compared to the least-squares approach.

The quality of reconstruction depends on the number of estimated principal components and therefore selection of the model rank is an important problem. In this section, we briefly outline possible solutions to this problem leaving out the detailed discussion.

The most straightforward way to define the rank of the model is to use cross-validation. The data set is divided into training and validation parts, models with different ranks are fitted to the training set and the the rank is selected based on the performance on the held-out validation set.

Another way is to use probabilistic methods for model selection. For example, Minka (2001) developed a means to detect the dimensionality of the principal subspace for *complete* data using Laplace's method and Bayesian information criterion (BIC) applied to the PPCA model. He also discussed other alternatives but his approach showed the best performance. However, the extension of Laplace's method to the case of *incomplete* data is not straightforward. An additional difficulty is the possibility that the posterior distribution over the model parameters has multiple modes corresponding to different principal subspaces. VBPCA can select the optimal number of components automatically by setting some columns of $\mathbf{W}$ to zero. However, the VB cost function may have many local minima which correspond to different model ranks and exploring different local solutions can be a tedious procedure. Hoff (2008) estimated the dimensionality of the principal subspace in the complete data using a sampling procedure.

6. Speeding Up Learning For Sparse High-Dimensional Data Sets

Obtaining a reasonably good solution in appropriate time is a very important issue for high-dimensional problems in which learning may take several days, as in the example considered in Section 7.4. The algorithms presented in the previous sections scale differently to problems with large dimensions and large degree of sparsity of observed values.

The computational complexities of different algorithms are summarized in Tables 2 and 3. For example, the alternating optimization algorithm for PPCA requires computation and inversion of matrices $\sum_{j\in O_i}\left[\mathbf{x}_j\mathbf{x}_j^{\mathrm{T}}+\boldsymbol{\Sigma}_{\mathbf{x}_j}\right]$ and $\left(v_y\mathbf{I}+\sum_{i\in O_j}\mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}\right)$, which are generally unique for each row of $\mathbf{W}$ and each column of $\mathbf{X}$, respectively. The corresponding computational complexity is $O(Nc^2+nc^3)$ per iteration,[2] which is quite a bit heavier than $O(ndc)$ per iteration (Roweis, 1998) required for complete data.

The computational complexity $O(Nc+nc)$ of the gradient method is small compared to previously reported results. For instance, Srebro and Jaakkola (2003) reported the complexity $O(ndc^2)$ for a more general model which was a huge improvement over the complexity $O(n^3c^3)$ of the algorithms proposed even earlier. Using the numbers from the Netflix problem considered in Section 7.4 ($N\approx 10^8$, $n\approx 480189$, $d=17770$, $c=30$), we get rough estimates of the differences in practice: Imputation algorithm $nd^2\approx 10^{14}$, (Srebro and Jaakkola, 2003) $ndc^2\approx 10^{12}$, alternating $\mathbf{W}$–$\mathbf{X}$ $Nc^2+nc^3\approx 10^{11}$, and gradient $Nc+nc\approx 10^{9.5}$.

Once the model has been learned and fixed, doing inference on a new sample is fast. Using the alternating algorithm, it requires only one E-step with $O(N_0c^2+c^3)$, where $N_0\leq d$ is the number of observed values in the new sample, while imputation and gradient algorithms require a number of iterations with $O(dc)$ and $O(N_0c)$ each.

Efficient memory usage is an important issue for sparse data sets. It is preferable that the amount of used memory scales linearly with the number of observed values regardless of the data matrix dimensionalities. In Table 2, we provide the number of parameters estimated by different models, which gives the lower bound for the amount of the required memory. Only the imputation algorithm

2. The computational complexity of the alternating $\mathbf{W}$–$\mathbf{X}$ scheme can be reduced if the matrices requiring inversion are computed once for all rows (columns) of $\mathbf{Y}$ that have observed values at exactly the same columns (rows respectively).

	Section	Number of parameters	Imputation with SVD	Alternating **W**–**X**	Gradient descent
LS	2	$(d+n)c$	✓	✓	✓
MAPPCA	6.4	$(d+n)c$		✓	✓
PPCA	3.1	$(d+n)c+nc^2$		✓	
VBPCA	3.3	$(d+n)(c+c^2)$		✓	
PPCAd	6.3	$(d+2n)c$			✓
VBPCAd	6.3	$(2d+2n)c$			✓

Table 2: Memory requirements and applicability of optimization methods to different models. We mark only the optimization schemes which can easily be implemented using the formulas provided in this paper. Some of the methods mentioned here will be presented in the upcoming sections.

Imputation with SVD	Alternating **W**–**X**	Gradient descent
$O(nd^2)$	$O(Nc^2+nc^3)$	$O(Nc+nc)$

Table 3: Summary of computational complexities (per iteration) of different optimization methods, assuming naïve computation of products and inverses of matrices and ignoring the computation of SVD in the imputation algorithm.

requires memory for the complete data matrix, and even that requirement can be diminished at a cost of greater time complexity.

6.1 Gradient-Based Learning

The gradient-based optimization scheme, extending the Oja learning algorithm to regularized PCA models, can be very efficient for large-scale problems and sparse data sets. The computational complexity of one iteration scales very well with dimensionalities, which can lead to faster learning in practice. The gradient-based approach can also be advantageous compared to the alternating optimization scheme as the latter discards the joint effect of parameters $\mathbf{W}$ and $\mathbf{X}$ on the changes in the cost function. This results in slow convergence without proper speed-up procedures (e.g., Honkela et al., 2003).

We also propose to use a speed-up to the gradient descent algorithm. In Newton's method for optimization, the gradient is multiplied by the inverse of the Hessian matrix. Newton's method is known to be fast-converging, but using the full Hessian is computationally costly in high-dimensional problems ($d \gg 1$). We propose a simplified approach which uses only the diagonal part of the Hessian matrix and includes a control parameter α that allows the learning algorithm to vary from the standard gradient descent ($\alpha = 0$) to the diagonal Newton's method ($\alpha = 1$). The final learning rules then take the form

$$\theta_i \leftarrow \theta_i - \gamma \left(\frac{\partial^2 C}{\partial \theta_i^2} \right)^{-\alpha} \frac{\partial C}{\partial \theta_i}.$$

For example, the cost function (4) has the second-order derivatives

$$\frac{\partial^2 C}{\partial w_{ik}^2} = \sum_{j \in O_i} x_{kj}^2, \qquad \frac{\partial^2 C}{\partial x_{kj}^2} = \sum_{i \in O_j} w_{ik}^2,$$

which can be computed efficiently for sparse data. We showed the efficiency of the proposed optimization scheme in the application to large-scale PCA problems in our conference papers (Raiko et al., 2008, 2007a).

6.2 Online Learning

In cases where the number of data samples is very large, $n \gg d$, it is wasteful to go through the whole data set before updating $\mathbf{W}$ (and $\mathbf{m}$). Using online learning (Oja, 1983), one sample j is processed at a time as follows: The principal components $\mathbf{x}_j$ are inferred using (8) (or the E-step in general), and $\mathbf{W}$ is updated using the gradient approach. When sweeping through the data, updates concerning the latter samples benefit from using the $\mathbf{W}$ that has already been updated using the earlier ones. This reduces the required number of iterations. Downsides of this approach include that determining the step size, or using speed-ups such as the one in Section 6.1 or the conjugate gradient, becomes difficult. Also, for enabling parallel implementation, one should process at least a small batch of samples at a time before updating $\mathbf{W}$ (e.g., Salakhutdinov et al., 2007). Online learning was not considered in this paper.

6.3 Factorial Variational Approximations

The overfitting problem can be more severe for high-dimensional sparse data sets. The situations discussed in Sections 2.4 and 3.2 are more probable when data points are sparsely distributed in high dimensions. Thus, methods penalizing large values of model parameters and taking into account their posterior uncertainty are especially relevant here.

The PPCA and VBPCA approaches, which take into account the posterior uncertainty at least in $\mathbf{X}$, are hardly applicable to large-scale problems. First, the computational burden of one iteration of the alternating $\mathbf{W}$–$\mathbf{X}$ scheme is very large (see Table 3) and application of the gradient-based optimization is cumbersome because of the need to compute $c \times c$ covariance matrices. Second, the required memory is at least nc^2 elements for storing posterior correlations only in $\mathbf{X}$. This becomes infeasible for many large-scale problems even for a decent number of principal components.

A possible solution is to take into account only some posterior correlations and to use variational techniques for learning. For example, the posterior approximation $q(\theta)$ in VBPCA can be made fully factorial leading to

$$q(\theta) = \prod_{i=1}^{d} q(m_i) \prod_{i=1}^{d} \prod_{k=1}^{c} q(w_{ik}) \prod_{c=1}^{k} \prod_{j=1}^{n} q(x_{kj}). \tag{38}$$

instead of (23). Such posterior approximation was used, for example, by Raiko et al. (2007b) for PCA in the presence of missing values. The implementation proposed there was based on the imputation algorithm and thus not easily scalable to high-dimensional sparse problems.

The fully factorial approximation (38) reduces significantly the number of variational parameters. They now include the mean parameters $\overline{m}_i$, $\overline{w}_{ik}$ and $\overline{x}_{kj}$ and the variance parameters $\widetilde{m}_i$, $\widetilde{w}_{ik}$, $\widetilde{x}_{kj}$ in addition to hyperparameters v_y, $v_{w,k}$, v_m, which can be point-estimated. The corresponding

cost function is given in Appendix D. It can be minimized in different ways, for example, by using the gradient-based optimization scheme explained in Section 6.1. The complete algorithm works by alternating four update steps: $\{\widetilde{w}_{ik}, \forall i,k\}$, $\{\widetilde{x}_{kj}, \forall k,j\}$, $\{\overline{w}_{ik}, \overline{x}_{kj}, \forall i,k,j\}$, and $\{v_y, v_{w,k}, v_m, \forall k\}$. The required derivatives are reported in Appendix D. We will further refer to this algorithm as *VBPCAd*.

The idea of fully factorial approximation can also be used for reducing the complexity of PPCA. The posterior of the hidden states $p(\mathbf{X}|\mathbf{Y},\mathbf{W},\mathbf{m},v_y)$ can be approximated to be fully factorial on the E-step. The approximating pdf (38) can be fitted to the true posterior by minimizing the cost function

$$C_{\mathrm{PPCA}} = \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{Y},\mathbf{X}|\mathbf{W},\mathbf{m},v_y)} d\mathbf{X},$$

which is motivated by the variational view of the EM algorithm. The resulting update rules resemble the ones of VBPCAd with the exceptions outlined in Fig. 5. We refer to this approach as *PPCAd*.

Note that the approximation used in PPCAd does not restrict the generality of the PPCA model when it is applied to *complete* data. The variational approximation introduces a bias in favor of solutions for which the form of the posterior approximation agrees with the form of the true posterior (e.g., Ilin and Valpola, 2005). The posterior covariance matrix $\mathbf{\Sigma}_{\mathbf{x}_j}$, which is the same for each j in the fully observed case, is diagonal if and only if the columns of $\mathbf{W}$ are mutually orthogonal. Therefore, restricting the posterior covariance to be diagonal guarantees that (26) holds at convergence. Requirements (24)–(25) are fulfilled because of the assumed prior model for $\mathbf{X}$. Thus, PPCAd for complete data should converge to the PCA basis. PPCAd applied to *incomplete* data is biased in favor of solutions in which the true posterior covariance matrices are closer to being diagonal. Note also that the idea of speeding up learning by using transformations as described in Section 4.3 for VBPCA, cannot be used for VBPCAd, because the transformations would break up the diagonality of the posterior covariance matrices.

There is an interesting connection between the speed-up proposed in Section 6.1 and the fully diagonal posterior approximation. The second order derivatives needed for the speed-up coincide with the inverse of the posterior variance of each parameter, and thus their computation is practically free.

6.4 Ignoring Posterior Uncertainty (MAPPCA)

A further simplification is to use maximum a posteriori (MAP) estimation for the model parameters. The minimized cost function is then minus log-posterior of the parameters. Assuming uniform prior for hyperparameters v_y, $v_{w,k}$, v_m, it is proportional to

$$C_{\mathrm{MAP}} = \frac{1}{v_y} \sum_{ij \in O} (y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i)^2 + N \log 2\pi v_y + \frac{1}{v_m} \sum_{i=1}^{d} m_i^2 + d \log 2\pi v_m \\ + \sum_{k=1}^{c} \left[\frac{1}{v_{w,k}} \sum_{i=1}^{d} w_{ik}^2 + d \log 2\pi v_{w,k} + \sum_{j=1}^{n} x_{kj}^2 + n \log 2\pi \right]. \quad (39)$$

and it should be minimized w.r.t. the unknown quantities $\mathbf{W}, \mathbf{X}, v_y, v_{w,k}, v_m$. We refer to this approach as MAPPCA.

Minimization of (39) can be done by any suitable optimization procedure. Derivatives required for gradient-based optimization are given in Appendix E. However, some difficulties should be

avoided. First, using an improper prior for hyperparameters leads to a situation where the posterior pdf is infinitely large when $v_{w,k} \to 0$, $\sum_{i=1}^{d} w_{ik}^2 \to 0$. This can be overcome by penalizing too small values of the variance hyperparameters. For example, a hyperparameter v_z, which is a variance parameter of the prior for a set of zero-mean variables $\{z_i, i = 1, \ldots, M\}$, can be updated as $v_z = \frac{2\beta + \sum_{i=1}^{M} \langle z_i^2 \rangle}{2\alpha + M}$, where $\langle z_i^2 \rangle$ denotes the expectation of z_i^2 over the posterior, and α and β are some small values (we used $\alpha = 10^{-3}$ and $\beta = 10^{-3}$). This update rule corresponds to broad priors restricting v_z to be positive and it guarantees that $v_z \geq \frac{\beta}{\alpha + M/2}$. The maximum likelihood (ML) estimate $v_z = \frac{1}{M} \sum_{i=1}^{M} \langle z_i^2 \rangle$ is achieved when $\alpha \to 0, \beta \to 0$.[3]

Another issue is the non-identifiability of scaling between $\mathbf{W}$ and $\mathbf{X}$. A practical way to avoid this problem is to fix the scale by normalizing the rows of $\mathbf{X}$ to zero mean and unit variance (after each update) and compensating by scaling the columns of $\mathbf{W}$ accordingly. This guarantees the fulfilment of (24)–(25), which are the conditions justified by the prior model.

6.5 Summary of the PCA Variants

The differences between the six described variants of the PCA models are summarized in Table 4 and Fig. 5. The approaches differ on the posterior models and on the use of prior for $\mathbf{W}$ and $\mathbf{m}$. All the variants can be derived as special cases of VBPCA by making simplifying assumptions, which is shown in Fig. 5.

7. Experiments

We study the properties of the discussed algorithms first using artificial data and then presenting a case study using the Netflix competition data (Netflix, 2007). In the experiments with artificial data, we generated data matrices $\mathbf{Y}$ according to model (12)–(14) with fixed dimensionalities d, c and n. The mixing matrix $\mathbf{W}$ was generated by taking a random orthogonal matrix and scaling its columns by $1, 2, \ldots, c$. The bias term $\mathbf{m}$ was selected randomly from a Gaussian distribution with variance 10. For one of the data sets (10-5-mlp), we nonlinearly transformed the resulting data. The observed values were selected randomly in $\mathbf{Y}$ and the rest of the data matrix was used to compute the test reconstruction error.

We used four types of data sets in the experiments:

- *Data Set 10-5-g*: The data set is for the case where $d \ll n$ and Gaussian components. We generated $n = 1000$ samples from a 10-dimensional Gaussian distribution with $c = 5$ principal directions. The standard deviation of the noise in (14) was $\sqrt{v_y} = 0.5$.

- *Data Set 10-5-mlp*: The data lie within a nonlinear manifold and $d \ll n$. The data were generated by a random nonlinear transformation of five-dimensional Gaussian data to the 10-dimensional space and adding relatively small Gaussian noise. The transformation was performed by randomly initialized multi-layer perceptron networks. The number of samples was $n = 1000$.

- *Data Set 10-4-mog*: The data set is a linear combination (12) of non-Gaussian $\mathbf{x}_j$ and $d \ll n$. We used $c = 4$ independently distributed non-Gaussian components (elements of $\mathbf{x}_j$). The

3. In the formulas reported in the article, we always present the ML update rules, although more robust update rules were used in practice.

	prior for $\mathbf{W}, \mathbf{m}$	posterior for $\mathbf{W}, \mathbf{m}$	posterior for $\mathbf{X}$
LS	uniform	point	point
PPCA	uniform	point	full
PPCAd	uniform	point	diagonal
VBPCA	yes	full	full
VBPCAd	yes	diagonal	diagonal
MAPPCA	yes	point	point

Table 4: Variants of probabilistic approaches to PCA.

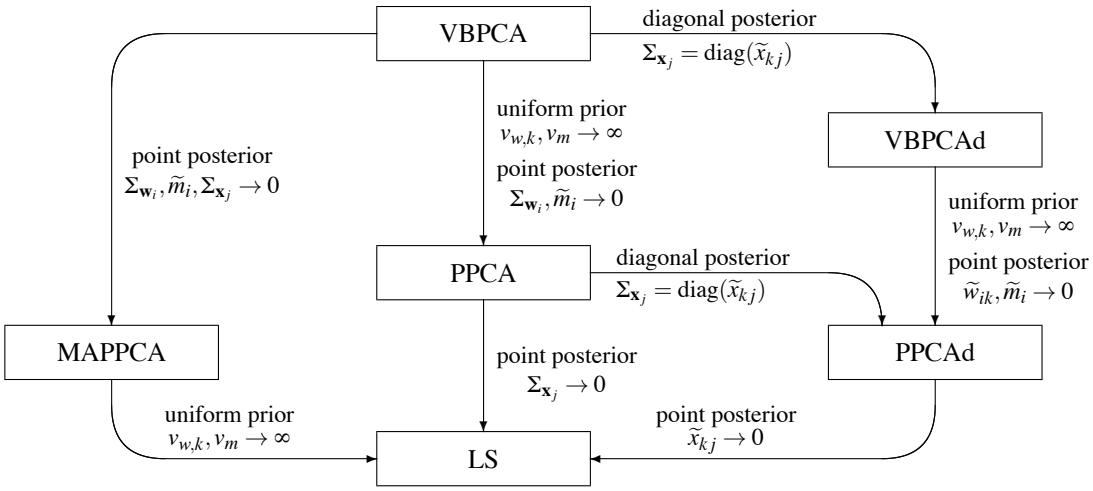


Figure 5: Relationships between the PCA variants are easily described by deriving each of them as a special case of VBPCA. The corresponding update rules can also be derived from the VBPCA update rules using the given restrictions.

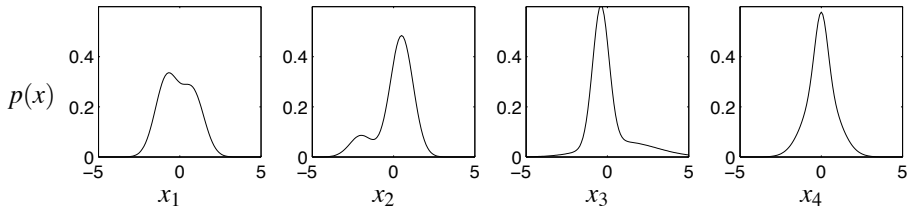


Figure 6: The distributions of the four components used to generate the 10-4-mog data set.

Data Set	d	n	c	Gaussianity	Square root of variances
10-5-g	10	1000	5	yes	[5 4 3 2 1 0.5 ... 0.5]
10-5-mlp	10	1000	5	no	varying by nonlinear transformation
10-4-mog	10	1000	4	no	[4 3 2 1 0.5 ... 0.5]
100-10-g	100	100	10	yes	[10 9 ... 1 0.1 ... 0.1]

Table 5: Characteristics of four types of data sets used in artificial experiments.

distributions of the four components are presented in Fig. 6. The standard deviation of the noise in (14) was $\sqrt{v_y} = 0.5$. The number of samples was $n = 1000$.

- *Data Set 100-10-g*: The data set is for the case where $d \approx n$ and small noise. The data were generated from a 100-dimensional Gaussian distribution with $c = 10$ principal directions. The standard deviation of the noise in (14) was $\sqrt{v_y} = 0.5$. The number of samples was $n = 100$.

The ratio of missing values was varied for all data sets. We used data sets with relatively small dimensionalities in order to test the performance of many alternative algorithms in a reasonable time. The summary of the data sets is shown in Table 5.

Different PCA algorithms considered in this article were used to estimate the same number c of principal components that was used for generating the data (see Table 5). The algorithms were initialized randomly except for the imputation algorithm which has a standard procedure of infilling missing values with the row-wise means of $\mathbf{Y}$.

The performance of the algorithms was assessed by the speed of their convergence and the accuracy in the task of missing value reconstruction. We computed three quantities: the average training root mean square (RMS) reconstruction error $\sqrt{\frac{1}{N}\sum_{ij\in O}(y_{ij} - \hat{y}_{ij})^2}$, the average test RMS reconstruction error and the average time to convergence. Averaging was done across 30 different data sets of the same type and the same ratio of missing values. The algorithms were run until the deviation of the training error was less than 0.0001 during 100 iterations. This stopping criterion was used to estimate the time to convergence.

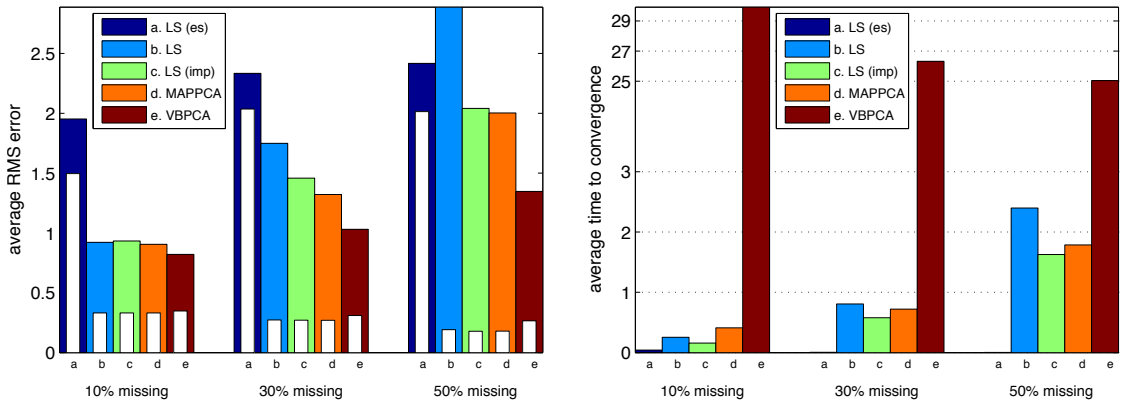
7.1 Performance of Least Squares Approaches and MAPPCA When $d \ll n$

In this section, we present the performance of three techniques:

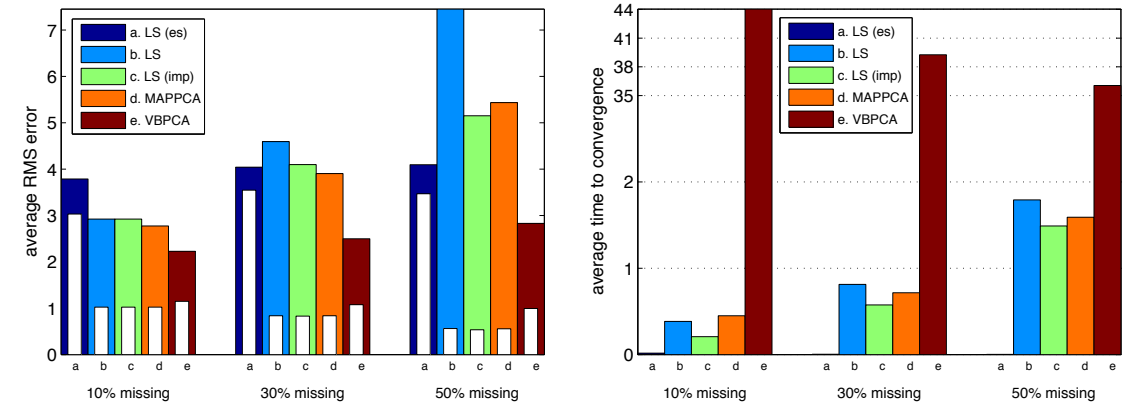
- LS: the least squares approach which optimizes the cost function (4) using the gradient-based procedure with the speed-up explained in Section 6.1
- LS (es): the same algorithm which also used early stopping: 20% of data was reserved as validation set and learning was stopped when the reconstruction error for the validation set started to increase
- LS (imp): the imputation algorithm presented in Section 2.5
- MAPPCA: method presented in Section 6.4. We used the gradient-based optimization procedure with the speed-up explained in Section 6.1.

The estimated performance of the four algorithms is presented in Fig. 7. There, we also show the results of VBPCA which turned out to be the most accurate method. We summarize the observed results in the following.

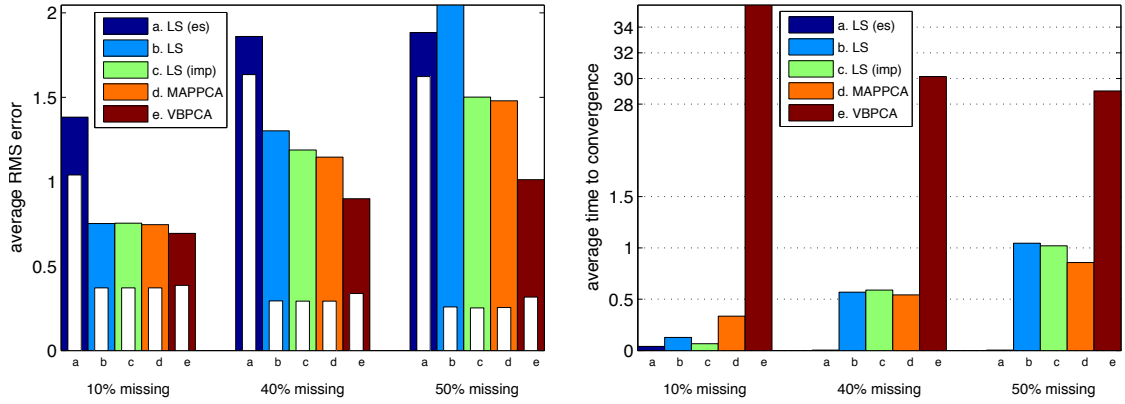
The gradient-based LS approach can be used for data sets with a relatively small amount of missing data. For such data sets, it provides reasonable accuracy, converges fast and scales well to high-dimensional data. However, its performance deteriorates fast with the increase of the ratio of missing values. The algorithm can get stuck at regions where learning proceeds very slowly and overfitting problems become very frequent. When overfitting happens, some parameter values can grow very large to explain perfectly only part of data. This results in very bad generalization and large test reconstruction errors (see discussion in Section 2.1). Thus, this method can badly overfit



(a) Data Set 10-5-g



(b) Data Set 10-5-mlp



(c) Data Set 10-4-mog

Figure 7: Left: Average test RMS errors (colored bars) and training RMS errors (white bars) obtained from 30 realizations of data sets of three types. Right: Average time to convergence, in seconds, estimated in the same way. Note the different scale on the r.h.s. plots for VBPCA.

even in relatively simple problems. We observed similar performance for the alternating optimization (see Section 2.2) of the sum-squared cost function: The alternating optimization procedure provided similar reconstruction errors but it seemed to get stuck in regions with slow convergence more often.

The simplest strategy of *early stopping* did not seem to improve the performance of the LS algorithms. On the contrary, the obtained accuracy was the worst among the considered approaches.

The imputation algorithm can also be a good choice for data sets with a relatively small amount of missing data. The algorithm converges fast, provides good accuracy and it has less problems with overfitting compared to explicit optimization of the sum-squared error. However, the complexity of the required computations grows fast with the increase of the dimensionalities, and therefore the algorithm is hardly applicable to large-scale data sets.

MAPPCA is a fast algorithm which provides reasonable accuracy for data sets with a relatively small amount of missing data. In many cases, it slightly outperforms the LS approaches in terms of the reconstruction accuracy. It is also less prone to serious overfitting problems compared to explicit optimization of (4). Again, similar performance was observed for MAPPCA optimized with the alternating optimization procedure explained in Section 2.2.

7.2 Performance of More Advanced Probabilistic Approaches When $d \ll n$

In this section, we present the performance of the more advanced probabilistic approaches:

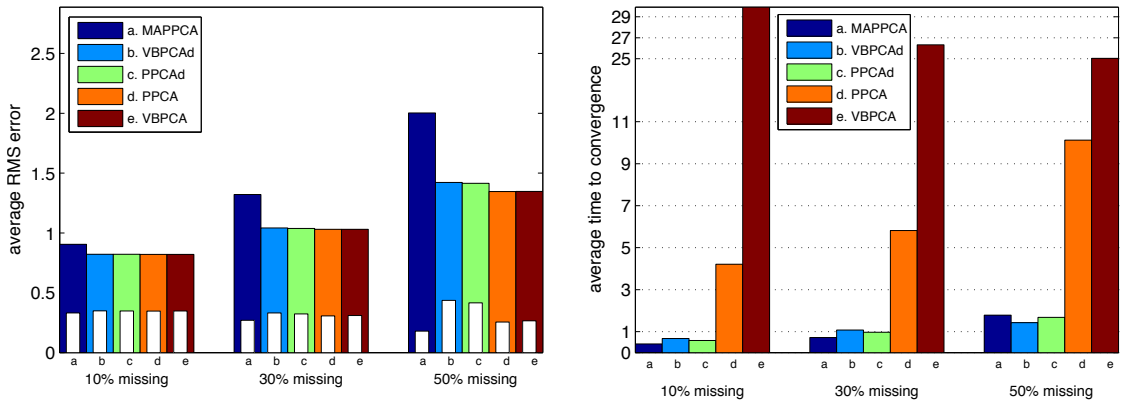
- PPCA presented in Section 3.1
- PPCAd which is PPCA with fully factorial variational approximation explained in Section 6.3
- VBPCA presented in Section 3.3
- VBPCAd which is VBPCA with fully factorial variational approximation explained in Section 6.3.

We used the gradient-based optimization procedure with the speed-up explained in Section 6.1 for training PPCAd and VBPCAd.

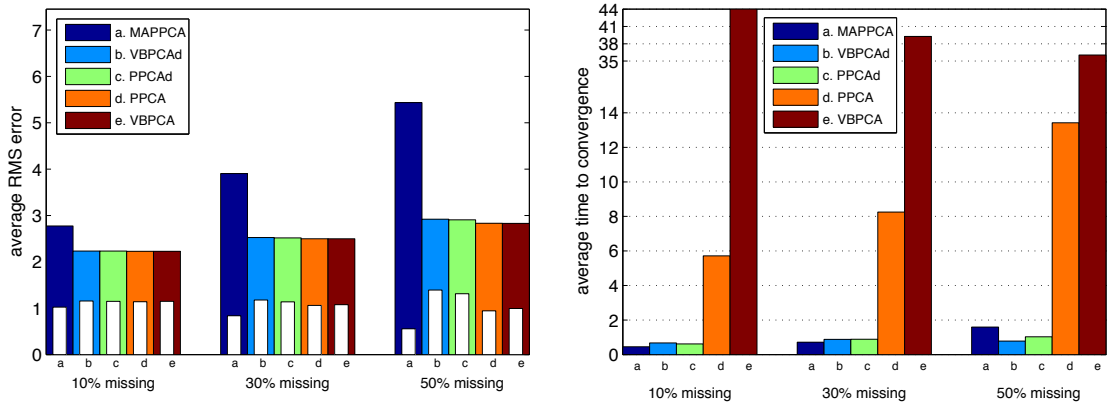
Fig. 8 presents the estimated performance of the algorithms for three data sets with $d \ll n$. We also show the results of MAPPCA for comparison. We summarize the results in the following.

Approaches using variational approximations, especially VBPCA and VBPCAd, may suffer from the underfitting problem. The algorithms may converge to a sub-optimal solution (corresponding to a local minimum of the VB cost function) in which some potentially useful principal components are not used. The algorithms seem to find such sub-optimal solutions more often for sparse data. Special care has to be taken to avoid such local minima. For example, in the presented experiments we fixed priors for $\mathbf{W}$ to be very broad at the beginning of learning and this helped improve the performance of VBPCA and VBPCAd significantly. See also the experiment showing the effect of the broad priors in Section 7.3.

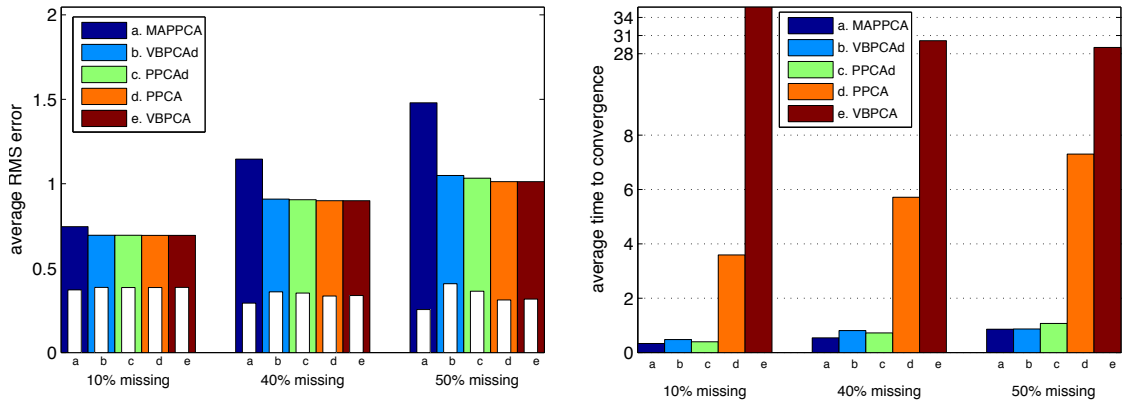
PPCAd is a very efficient algorithm for data sets with $d \ll n$. Its reconstruction accuracy is comparable to the best results (obtained with VBPCA) but the algorithm converges fast and scales well to high-dimensional problems. The difference of the accuracy of PPCAd compared to the best results of VBPCA becomes more noticeable when the ratio of missing values or the number of estimated principle components increase.



(a) Data Set 10-5-g



(b) Data Set 10-5-mlp



(c) Data Set 10-4-mog

Figure 8: Left: Average test RMS errors (colored bars) and training RMS errors (white bars) obtained from 30 realizations of data sets of three types. Right: Average time to convergence, in seconds, estimated in the same way. Note the different scale on the r.h.s. plots for VBPCA.

VBPCAd provides similar performance to PPCAd for data sets with $d \ll n$ but it is more prone to the underfitting problem mentioned earlier.

VBPCA and *PPCA* are the most accurate algorithms in these experiments. However, they are also very slow because of the required computations of the posterior covariance matrices. Thus, the methods are computationally very expensive to use in high-dimensional problems but if they are feasible, they are worth trying. The difference in the accuracy of VBPCA and PPCA is not noticeable in these experiments. This is likely due to the fact that the number n of samples was very large and the effect of priors was very small. VBPCA is also more prone to the underfitting problem compared to PPCA.

7.3 Performance in Difficult Sparse Problems

In this section, we test the performance of the considered algorithms on a more challenging data set 100-10-g. The data set is quite sparse (up to 75% of values are missing) and the number n of samples is very small compared to the data dimensionality d. Sparse data sets with $d \approx n$ can appear in some practical applications. For example, in collaborative filtering the number of users can be comparable to the number of ranked items and the number of missing values can be large. Here we used Gaussian data with $n = d = 100$, $c = 10$ principal directions and with relatively small noise.

Fig. 9 presents the performance of the considered algorithms for these data. The LS algorithms and MAPPCA do not provide good performance because of overfitting problems. The probabilistic approaches perform much better and the best results are obtained with VBPCA. The advantage of VBPCA becomes more significant for data sets with many missing values. The experiments also show a larger effect of using priors in VBPCA and VBPCAd (in comparison with PPCA and PPCAd) for data sets with a relatively small number of samples.

One of the reasons for the superior performance of the VB approach is its ability to select automatically the optimal rank of the model by cutting off useless principal components. Model selection is useful even in such artificial tests (when the number of principal components used for data generation is known) because the number of components which are useful for reliable reconstruction of missing data can turn out to be smaller than the model rank used for generating the data. Thus, the reported accuracy of the algorithms alternative to VBPCA and VBPCAd might be improved if model selection was explicitly performed for them (e.g., by cross-validation).

In Section 7.2, we pointed out that the downside of the VB model selection is the existence of multiple sub-optimal solutions corresponding to different local minima of the VB cost function. Solutions in which the effect of some components is set to zero are potentially attractive for the method. In order to avoid sub-optimal solutions, we fixed the priors for $\mathbf{W}$ to be very broad at the beginning of learning for VBPCA and VBPCAd in our experiments. We show the effect of the number of iterations with fixed broad priors on the resulting accuracy in Fig. 10.

A useful feature of the VB approach is that the value of the VB cost function at a local minimum can be used to estimate the accuracy of the corresponding model. This follows from the fact that the VB cost gives the lower bound of the likelihood, as shown in (22). In Fig. 11, we demonstrate the correlations between the accuracy of a trained VBPCA model and the corresponding value of the cost function. In that experiment, we initialized VBPCA in two different ways and applied it to 30 realizations of the 100-10-g data set with 80% of missing data. The negative result was that different initializations led to different VBPCA solutions for most of the 30 realizations. The positive result

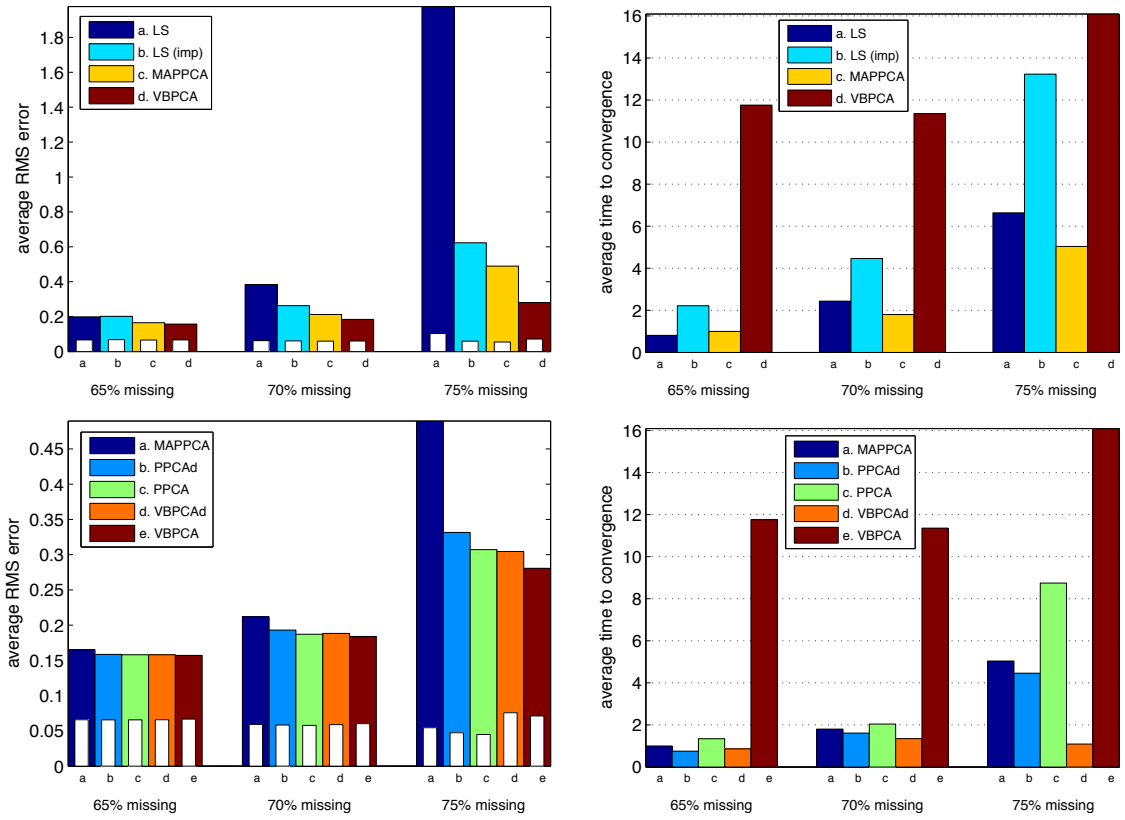


Figure 9: Left: Average test rms errors (colored bars) and training rms errors (white bars) obtained from 30 realizations of data set 100-10-g. Right: Average time to convergence, in seconds, estimated in the same way.

was that the solutions with the smaller cost function values generally provided better reconstruction accuracy. Similar results were obtained for VBPCAd (not shown here).

Thus, in order to find a *globally* good VB solution, one could, in practice, run the learning algorithm many times. The algorithm should be run until *full convergence*, otherwise the cost function values cannot be used for model comparison. Running VBPCA can be computationally heavy because it is the slowest algorithm among the considered ones. Therefore, VBPCAd is an attractive alternative for large-scale data sets.

7.4 Experiments with the Netflix Data

We have tested different PCA approaches in the task of collaborative filtering. The Netflix (2007) problem is a collaborative filtering task that has received a lot of attention recently. It consists of movie ratings given by $n = 480189$ customers to $d = 17770$ movies. There are $N = 100480507$ ratings from 1 to 5 given, and the task is to predict 2817131 other ratings among the same group of customers and movies. 1408395 of the ratings are reserved for validation (or probing). Note that 98.8% of the values are thus missing.

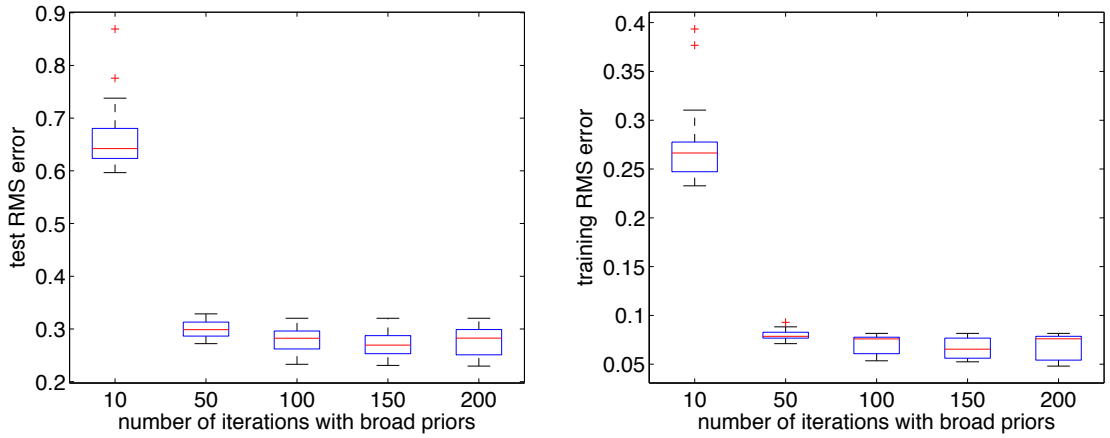


Figure 10: The underfitting effect for VBPCA when the priors are updated too early: Box plots of test (left) and training (right) RMS errors depending on the number of iterations with fixed broad priors for $\mathbf{W}$ (x-axes) at the beginning of learning. VBPCA was applied to 30 data sets of type 100-10-g with 75% of missing values.

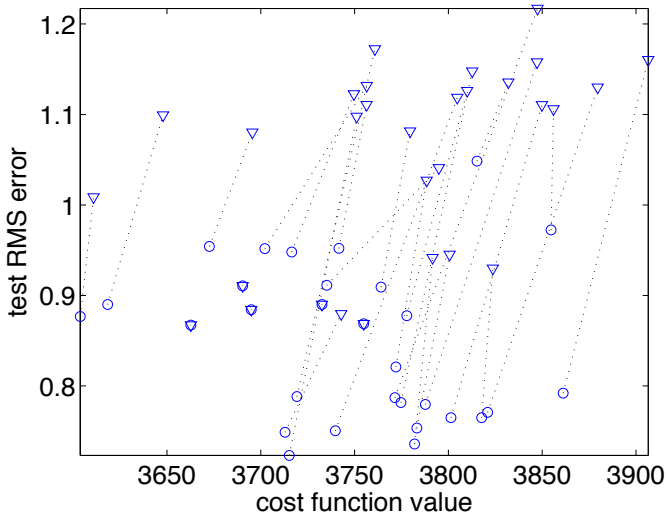


Figure 11: Test RMS error (y-axis) against cost function values (x-axis) obtained with VBPCA for 30 realizations of the data set 100-10-g with 80% of missing values. Two initializations were used for each realization: random (the corresponding local minima are shown with triangles) and with 100 iterations of PPCAd (the corresponding local minima are shown with circles). The dotted lines connect two solutions obtained for the same realization.

The PCA approach is one of the most popular techniques considered by the Netflix contestants (e.g., Funk, 2006; Bell et al., 2007; Salakhutdinov et al., 2007; Paterek, 2007; Lim and Teh, 2007). In our recent conference papers (Raiko et al., 2007a, 2008), we tried to estimate $c = 15$ principal components from the data using unregularized (LS) PCA, MAPPCA, and VBPCAd approaches. We reproduce the results for the LS approaches in Fig. 12. The results clearly show that overfitting is a serious problem for this sparse data set and the LS approach fails. It is also evident that the alternating optimization algorithm is very slow and it can hardly be used for such high-dimensional data. The imputation algorithm is also infeasible in this problem (Raiko et al., 2008). The proposed remedy is to use probabilistic approaches.

Fig. 13 presents the results obtained with MAPPCA and VBPCAd using a varying number of principal components. MAPPCA starts to overfit after about five hours of computation, that is, the RMS error on the test data (lower curves) starts to degrade even if the RMS error on the training data is still diminishing. This does not happen with VBPCAd as the validation error seems to decrease monotonically. The experiments also confirm that VBPCAd is computationally scalable to very large problems. We also tried to run VBPCA, but the straightforward Matlab implementation turned out to be too slow to produce meaningful results. The experiments were run on a dual cpu AMD Opteron SE 2220 using Matlab and the implementations did not use parallel computing.

Our best RMS error for the probing Netflix data was 0.9055 and it was achieved with VBPCAd with 50 components. The same model re-trained using both training and probing data provided an RMS error 0.8990 on the quiz set. The only pre/post-processing that we used was bounding the reconstructions $\hat{y}_{ij}$ such that $1 \leq \hat{y}_{ij} \leq 5$. This is a good result compared to conceptually similar models developed by other contestants. For example, Bell et al. (2007) reported an RMS error of 0.9135 with basic factor analysis with 40 components. Lim and Teh (2007) studied a model which basically implements VBPCA and they reported an RMS error of 0.9141 for a model with 30 components. Salakhutdinov and Mnih (2008b) reported the probing RMS error 0.9280 for an SVD method which essentially implements the LS approach. Their constrained probabilistic matrix factorization model provided a probing RMS error 0.9016 but it used extra informations and more sophisticated priors. The accuracy of our VBPCAd model is comparable to the accuracy of an MCMC approach applied to a similar generative model (Salakhutdinov and Mnih, 2008a). The RMS error achieved with a model with 60 components trained by MCMC was 0.8989 for the quiz set.

Besides reconstruction of missing values, probabilistic PCA methods provide extra information which can be useful in practice. For example, one can predict the uncertainty of the provided reconstructions, as discussed in Section 5. Fig. 14 shows good correlation between the probing RMS errors and the uncertainty of the reconstructions computed by VBPCAd. It is clear that the predicted uncertainty is underestimated, which is largely because of the approximations used in modeling the posterior distributions.

The estimated uncertainty can provide additional information when PCA is used as a preprocesing step for more sophisticated methods. For example, Bell and Koren (2007) used PCA results for constructing a user-oriented neighborhood model for collaborative filtering. Because each column of matrix $\mathbf{X}$ computed by PCA can be considered as a collection of some features associated with a particular user, the similarity between n users can be estimated based on the similarity between the columns of $\mathbf{X}$. The information about the uncertainty of the feature estimates could be used to build a robust similarity measure: Users who have rated few movies (and whose features

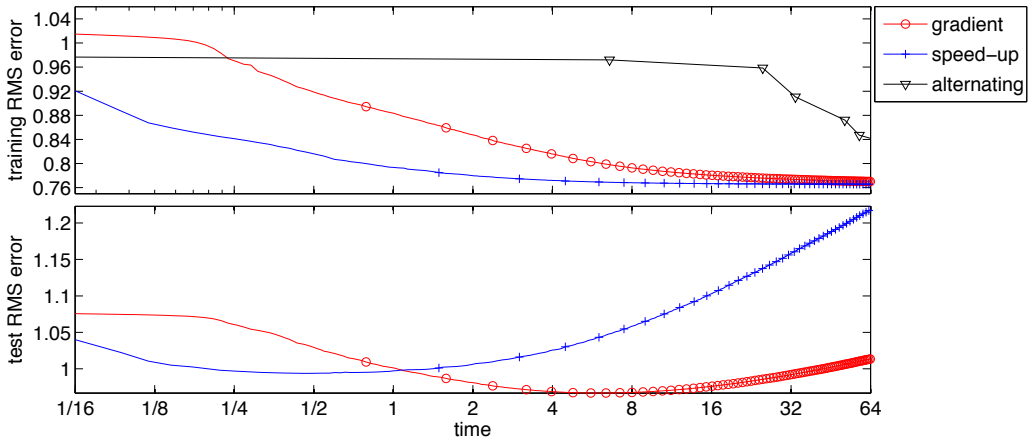


Figure 12: Learning curves for LS algorithms in the Netflix problem. The RMS error (y-axis) is plotted against the processor time in hours. The upper curves show the training error while the lower curves show the test error (for the probing data provided by Netflix). Note that the time scale is logarithmic. The probing error provided by the alternating algorithm is not shown because it is too large compared to the other two curves. The two gradient-based approaches were implemented using parallel computing (this implementation was about twice as fast as the implementation on a single CPU).

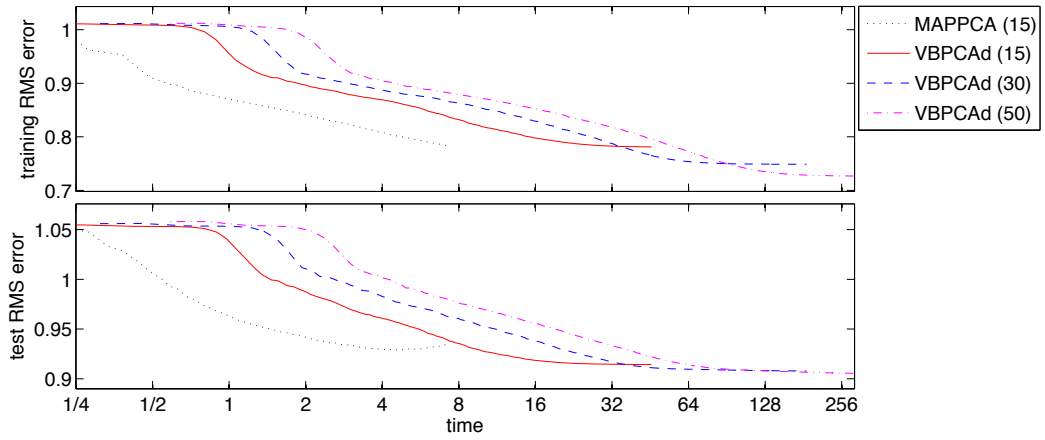


Figure 13: Learning curves for regularized methods in the Netflix problem. The RMS error (y-axis) is plotted against the processor time in hours. The upper curves show the training error while the lower curves show the test error (for the probing data provided by Netflix). Note that the time scale is logarithmic.

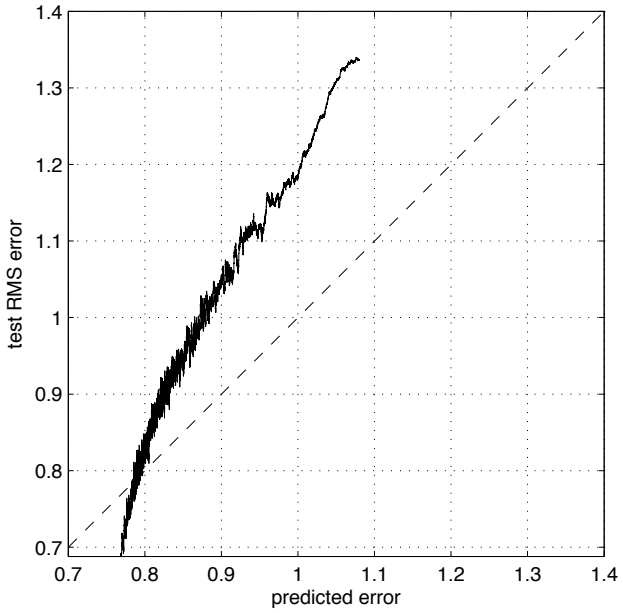


Figure 14: Uncertainty predicted by VBPCAd for the Netflix probing data. The RMS error for the probing data (y-axis) is plotted against the square root of the reconstruction variance estimated by VBPCAd with 50 components (x-axis). To show the dependency more clearly, the estimates of the probing RMS errors are obtained by averaging over a batch of ratings with similar predicted uncertainties.

are estimated with high uncertainty) should not affect the predictions of their neighbors as much as users who have rated many movies (and whose features are estimated more accurately).

Finally, we note that the accuracy achieved by the best teams in the Netflix competition is significantly better than the numbers reported here. However, that does not mean that PCA (and its implementations discussed in this paper) is of no use in collaborative filtering tasks. On the contrary, PCA or its close relatives were used by the leading teams (e.g., Bell et al., 2007) as an important element of the final blend of models. The key to success in that competition was in combining a vast collection of different models rather than in perfecting a single approach (e.g., The Ensemble, 2009).

8. Discussion

We have reviewed the problem of PCA learning in the presence of missing values and discussed various approaches to it. We demonstrated that the simplicity of PCA is lost when introducing missing values. Firstly, the estimation of the bias term and the covariance matrix of the data becomes difficult and thus the solution by eigen-decomposition cannot be used directly. Secondly, the convergence to a unique solution cannot be guaranteed even for the simplest PCA models.

Missing values also make the problem of overfitting more relevant to PCA, so there is a need for some form of regularization. In regularized solutions, reconstructions of data vectors are no longer

conditions	small-scale problems	large-scale problems
few missing values	imputation	gradient-based LS
$d \ll n$	PPCA	PPCAd
many missing values or $d \approx n$	VBPCA	VBPCAd

Table 6: Recommendations on the use of different PCA algorithms.

the projections of the data to the principal subspace. Regularization can be done elegantly using probabilistic models.

Inference and learning in the relevant probabilistic models are generally more complex compared to the complete data case. For example, the posterior covariance of principal components $\Sigma_{\mathbf{x}_j}$ is no longer same for each sample. Thus, a diagonal posterior covariance $\Sigma_{\mathbf{x}_j}$ is no longer sufficient for finding the optimal PCA solution. Regularized models typically have more local minima of the optimized cost function. In particular, there can be local minima corresponding to zero values of hyperparameters.

Another type of treatment for random variables is to use MCMC methods. Their benefits compared to VBPCA includes not having to assume posterior independence like in (23) and being able to vary the number of components (Hoff, 2008), but the downsides include worse interpretability, higher computational complexity, and the need to store samples in case the model is to be applied to new incoming data. Salakhutdinov and Mnih (2008a) used Gibbs sampling initialized with the MAPPCA method for the Netflix problem. The resulting accuracy was similar to ours with similar number of components (0.8989 with 60 components against our 0.8990 for VBPCAd with 50 components).

Large-scale PCA problems require fast converging learning algorithms which allow for efficient implementation. Additionally, sparse data sets require that the amount of computer memory would scale linearly with the number of observed values regardless of the data matrix dimensionalities. In this paper, efficient solutions are proposed based on fully factorial variational approximations and approximate Newton's iteration for the relevant optimization procedure. An alternative learning algorithm based on natural gradient was discussed in our conference paper (Raiko et al., 2008).

The choice of the right PCA algorithm depends on a particular data analysis problem. We gave some recommendations in Sections 7.1–7.3 and we summarize our recommendations in Table 6. The LS approaches can be applied to data sets with relatively few missing data: The *imputation* algorithm can be a good choice for smaller-scale problems and the gradient-based LS algorithms would be good for large data sets. When the number n of samples is relatively large and the data matrix is relatively densely populated, PPCA is a proper choice (or PPCAd for large-scale problems). VB algorithms can be most efficient when there are very few samples ($d \approx n$) or for very sparse data sets. Again, VBPCAd would be a better alternative for large-scale problems.

A Matlab toolbox which contains implementations of all the considered PCA techniques is available online at `http://www.cis.hut.fi/projects/bayes/`. Some of the implemented algorithms scale well to large-scale sparse data sets, which is achieved by low level implementation of core computations and by support of parallel computing. For example, the experiments with the Netflix data reported in Fig. 13 were obtained using the provided Matlab code.

Acknowledgments

The authors would like to thank Juha Karhunen, Erkki Oja, Alexey Kaplan, and Jaakko Luttinen for useful discussions. This work was supported by the Academy of Finland (projects 'Unsupervised machine learning in latent variable models', 'Novel machine learning techniques for studying climate variability and its impacts', 'Auditory approaches to automatic speech recognition' and 'Bayesian Machine Learning Algorithms for Discovering Relations with Latent Variable Models'), and by the IST Program of the European Community, under the PASCAL2 Network of Excellence. This publication only reflects the authors' views. We would like to thank Netflix (2007) for providing the data.

Appendix A. Notes on the Imputation Algorithm

The imputation algorithm can be seen as the application of the EM algorithm (Dempster et al., 1977) to the model

$$\mathbf{y}_j = \mathbf{W}\mathbf{x}_j + \mathbf{m} + \varepsilon_j$$

with isotropic Gaussian noise $p(\varepsilon_j) = \mathcal{N}(\varepsilon_j; 0, v_y\mathbf{I})$ and model parameters $\mathbf{W}$, $\mathbf{x}_j$, $\mathbf{m}$, v_y. Note that $\mathbf{x}_j$ belong to the model parameters and missing values in $\mathbf{y}_j$ are treated as *hidden variables*. This is contrary to the probabilistic PCA model (see Section 3.1) where $\mathbf{x}_j$ are treated as hidden variables. Note also that with this interpretation, there is no well defined inference procedure for a new data case outside of the training set.

Following the view of the EM algorithm presented by Neal and Hinton (1999), we can write down the cost function

$$\begin{aligned} C_{\text{imp}}(\theta, v_y, q(Y_{\text{mis}})) &= \int q(Y_{\text{mis}}) \log \frac{q(Y_{\text{mis}})}{p(\mathbf{Y}|\theta, v_y)} dY_{\text{mis}} \\ &= \int q(Y_{\text{mis}}) \log \frac{q(Y_{\text{mis}})}{p(Y_{\text{mis}}|\theta, v_y)} dY_{\text{mis}} - \log p(Y_{\text{obs}}|\theta, v_y)\,, \end{aligned} \tag{40}$$

which should be minimized w.r.t. model parameters $\theta = \{\mathbf{W}, \mathbf{m}, \mathbf{x}_j, \forall j\}$, v_y and the pdf over the missing data $q(Y_{\text{imp}})$. We denoted the missing data by $Y_{\text{mis}} = \{y_{ij}\,, ij \notin O\}$ and the observed data by $Y_{\text{obs}} = \{y_{ij}\,, ij \in O\}$.

Minimization of (40) w.r.t. $q(Y_{\text{mis}})$ for fixed θ and v_y yields that $q(Y_{\text{mis}}) = p(Y_{\text{mis}}|\theta, v_y)$ because the first term in (40) is simply the Kullback-Leibler divergence between the two pdfs. It is straightforward that

$$p(Y_{\text{mis}}|\theta, v_y) = \prod_{ij \in O} \mathcal{N}(y_{ij}; \hat{y}_{ij}(\theta), v_y)\,,$$

where $\hat{y}_{ij}(\theta)$ are the reconstructed missing values using (5) with the current estimates θ. Thus, infilling missing data with the reconstructions $\hat{y}_{ij}(\theta)$ is the step of the imputation algorithm which corresponds to the E-step of the EM-algorithm.

Collecting the terms in (40) which depend only on θ and v_y gives

$$\begin{aligned} &-\int q(Y_{\text{mis}}) \log p(Y_{\text{obs}}, Y_{\text{mis}}|\theta, v_y) dY_{\text{mis}} \\ &= -\frac{dn}{2}\log 2\pi v_y - \frac{1}{2v_y}\sum_{ij\in O}\left(y_{ij}-\hat{y}_{ij}(\theta)\right)^2 - \frac{1}{2v_y}\sum_{ij\notin O}\left[\left(\bar{y}_{ij}-\hat{y}_{ij}(\theta)\right)^2 + v_y\right] \\ &= -\frac{dn}{2}\log 2\pi v_y - \frac{1}{2v_y}\sum_{ij}\left(y_{ij}-\hat{y}_{ij}(\theta)\right)^2 - \frac{dn-N}{2}. \end{aligned} \tag{41}$$

It is easy to see that minimization of (41) w.r.t. θ is equivalent to minimizing the LS cost (2) for the infilled data matrix. This is done in the imputation algorithm by SVD of the complete data matrix. Thus, the M-step of the EM-algorithm corresponds to performing SVD in the imputation algorithm. Minimization of (41) w.r.t. v_y does not affect the steps of the imputation algorithm.

The imputation algorithm implicitly minimizes the cost function (4), thus it belongs to the class of LS PCA algorithms. It can be shown (Neal and Hinton, 1999) that the minimum of (40) coincides with the minimum of minus log-likelihood

$$-\log p(Y_{\text{obs}}|\theta, v_y) = \frac{N}{2}\log 2\pi v_y + \frac{1}{2v_y}\sum_{ij\in O}\left(y_{ij}-\hat{y}_{ij}(\theta)\right)^2. \tag{42}$$

Replacing v_y in (42) with its maximum likelihood estimate yields that the function being minimized is

$$\frac{N}{2}\left[\log\left(\frac{2\pi}{N}\sum_{ij\in O}\left(y_{ij}-\hat{y}_{ij}(\theta)\right)^2\right)+1\right],$$

from which it follows that the imputation algorithm implicitly optimizes the sum-squared error.

Appendix B. Conditions Fulfilled at the Convergence of Probabilistic PCA

The variational view of the EM algorithm allows for an interpretation of the PPCA learning algorithm in which the cost function

$$C = \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{X})} d\mathbf{X} - \int q(\mathbf{X}) \log p(\mathbf{Y}|\mathbf{W}, \mathbf{X}, \mathbf{m}, v_y) d\mathbf{X} \tag{43}$$

is minimized w.r.t. to the model parameters $\mathbf{W}$, $\mathbf{m}$, v_y and the pdf $q(\mathbf{X})$. The E-step fixes the model parameters and minimizes the cost function w.r.t. the distribution $q(\mathbf{X})$ while the M-step minimizes C w.r.t. $\mathbf{W}$, $\mathbf{m}$ and v_y assuming that $q(\mathbf{X})$ is fixed.

Let us consider simple transformations of $\{\mathbf{W}, \mathbf{m}, q(\mathbf{X})\}$ which do not change the second term in (43). For example, subtraction of a constant $\mathbf{b}$ from the columns of $\mathbf{X}$ can be compensated by changing the bias term $\mathbf{m}$ correspondingly:

$$q(\mathbf{X}) \leftarrow \prod_{j=1}^{n} \mathcal{N}\left(\mathbf{x}_j; \bar{\mathbf{x}}_j - \mu, \Sigma_{\mathbf{x}_j}\right), \tag{44}$$

$$\mathbf{m} \leftarrow \mathbf{m} + \mathbf{W}\mu. \tag{45}$$

because $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{m} = \mathbf{W}(\mathbf{x} - \mu) + (\mathbf{W}\mu + \mathbf{m})$. Similarly, rotation of the columns of $\mathbf{X}$ can be compensated by changing $\mathbf{W}$:

$$q(\mathbf{X}) \leftarrow \prod_{j=1}^{n} \mathcal{N}\left(\mathbf{x}_j; \mathbf{A}\overline{\mathbf{x}}_j, \mathbf{A}\Sigma_{\mathbf{x}_j}\mathbf{A}^{\mathrm{T}}\right), \tag{46}$$

$$\mathbf{W} \leftarrow \mathbf{W}\mathbf{A}^{-1} \tag{47}$$

because $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{m} = (\mathbf{W}\mathbf{A}^{-1})(\mathbf{A}\mathbf{x}) + \mathbf{m}$.

These transformations generally affect the first term in (43) which is the Kullback-Leibler divergence between $q(\mathbf{X})$ and $p(\mathbf{X})$:

$$D = \sum_{j=1}^{n} \int q(\mathbf{x}_j) \log \frac{q(\mathbf{x}_j)}{p(\mathbf{x}_j)} d\mathbf{x}_j = \frac{1}{2} \sum_{j=1}^{n} \left[\mathrm{tr}(\Sigma_{\mathbf{x}_j}) + \overline{\mathbf{x}}_j^{\mathrm{T}} \overline{\mathbf{x}}_j - \log\det \Sigma_{\mathbf{x}_j} \right]. \tag{48}$$

and at the convergence this term cannot be made smaller by transformations of the form (44)–(45) or (46)–(47).

Transformation (44)–(45) changes (48) to

$$D = \frac{1}{2} \sum_{j=1}^{n} \left[\mathrm{tr}(\Sigma_{\mathbf{x}_j}) + (\overline{\mathbf{x}}_j - \mu)^{\mathrm{T}} (\overline{\mathbf{x}}_j - \mu) - \log\det \Sigma_{\mathbf{x}_j} \right].$$

Now taking the derivative w.r.t. μ and equating it to zero gives the optimal $\mu = \frac{1}{n}\sum_{j=1}^{n} \overline{\mathbf{x}}_j$. At the convergence, the optimal μ should be zero and therefore (34) should hold.

Similarly, transformation (46)–(47) gives

$$\begin{aligned} D &= \frac{1}{2} \sum_{j=1}^{n} \left[\mathrm{tr}(\mathbf{A}\Sigma_{\mathbf{x}_j}\mathbf{A}^{\mathrm{T}}) + \overline{\mathbf{x}}_j^{\mathrm{T}} \mathbf{A}^{\mathrm{T}} \mathbf{A} \overline{\mathbf{x}}_j - \log\det(\mathbf{A}\Sigma_{\mathbf{x}_j}\mathbf{A}^{\mathrm{T}}) \right] \\ &= \frac{n}{2} \mathrm{tr}(\mathbf{A}\Sigma_* \mathbf{A}^{\mathrm{T}}) - \frac{1}{2} \sum_{j=1}^{n} \log\det(\mathbf{A}\Sigma_{\mathbf{x}_j}\mathbf{A}^{\mathrm{T}}), \end{aligned}$$

where $\Sigma_* = \frac{1}{n}\sum_{j=1}^{n} \left[\overline{\mathbf{x}}_j \overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j} \right]$. Taking the derivative w.r.t. $\mathbf{A}$ gives

$$\mathbf{A}\Sigma_* - (\mathbf{A}^{\mathrm{T}})^{-1} = 0$$

and therefore the optimal $\mathbf{A}$ satisfies $\mathbf{A}\Sigma_*\mathbf{A}^{\mathrm{T}} = \mathbf{I}$. Transformation (46)–(47) yields that $\Sigma_* \leftarrow \mathbf{A}\Sigma_*\mathbf{A}^{\mathrm{T}}$ and therefore at the convergence $\Sigma_* = \mathbf{I}$, which is condition (35).

Appendix C. Variational Bayesian PCA (VBPCA)

The following form of the posterior approximation is used:

$$q(\mathbf{W}, \mathbf{X}, \mathbf{m}) = \prod_{i=1}^{d} \mathcal{N}(m_i; \overline{m}_i, \widetilde{m}_i) \prod_{i=1}^{d} \mathcal{N}(\mathbf{w}_i; \overline{\mathbf{w}}_i, \Sigma_{\mathbf{w}_i}) \prod_{j=1}^{n} \mathcal{N}\left(\mathbf{x}_j; \overline{\mathbf{x}}_j, \Sigma_{\mathbf{x}_j}\right).$$

The update of the principal components:

$$\Sigma_{\mathbf{x}_j} = v_y\Big(v_y\mathbf{I} + \sum_{i\in O_j}\left[\overline{\mathbf{w}}_i\overline{\mathbf{w}}_i^{\mathrm{T}} + \Sigma_{\mathbf{w}_i}\right]\Big)^{-1},$$

$$\overline{\mathbf{x}}_j = \frac{1}{v_y}\Sigma_{\mathbf{x}_j}\sum_{i\in O_j}\overline{\mathbf{w}}_i(y_{ij}-\overline{m}_i), \qquad j = 1,\ldots,n.$$

The update of the bias term and matrix $\mathbf{W}$:

$$\overline{m}_i = \frac{v_m}{|O_i|(v_m + v_y/|O_i|)}\sum_{j\in O_i}[y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}}\overline{\mathbf{x}}_j], \tag{49}$$

$$\widetilde{m}_i = \frac{v_y v_m}{|O_i|(v_m + v_y/|O_i|)}, \tag{50}$$

$$\Sigma_{\mathbf{w}_i} = v_y\Big(v_y\,\mathrm{diag}(v_{w,k}^{-1}) + \sum_{j\in O_i}\left[\overline{\mathbf{x}}_j\overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j}\right]\Big)^{-1},$$

$$\overline{\mathbf{w}}_i = \frac{1}{v_y}\Sigma_{\mathbf{w}_i}\sum_{j\in O_i}\overline{\mathbf{x}}_j(y_{ij}-\overline{m}_i), \qquad i = 1,\ldots,d$$

and the variance parameters:

$$v_y = \frac{1}{N}\sum_{ij\in O}\left[(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}}\overline{\mathbf{x}}_j - \overline{m}_i)^2 + \widetilde{m}_i + \overline{\mathbf{w}}_i^{\mathrm{T}}\Sigma_{\mathbf{x}_j}\overline{\mathbf{w}}_i + \overline{\mathbf{x}}_j^{\mathrm{T}}\Sigma_{\mathbf{w}_i}\overline{\mathbf{x}}_j + \mathrm{tr}(\Sigma_{\mathbf{x}_j}\Sigma_{\mathbf{w}_i})\right],$$

$$v_{w,k} = \frac{1}{d}\sum_{i=1}^{d}\left(\overline{w}_{ik}^2 + \widetilde{w}_{ik}\right), \qquad v_m = \frac{1}{d}\sum_{i=1}^{d}\left(\overline{m}_i^2 + \widetilde{m}_i\right), \tag{51}$$

where $\widetilde{w}_{ik}$ is the k-th element on the diagonal of $\Sigma_{\mathbf{w}_i}$.

Appendix D. VBPCA with Fully Factorial Approximation (VBPCAd)

VBPCAd uses the fully factorial posterior approximation

$$q(\mathbf{W},\mathbf{X},\mathbf{m}) = \prod_{i=1}^{d}\mathcal{N}(m_i;\overline{m}_i,\widetilde{m}_i)\prod_{i=1}^{d}\prod_{k=1}^{c}\mathcal{N}(w_{ik};\overline{w}_{ik},\widetilde{w}_{ik})\prod_{k=1}^{c}\prod_{j=1}^{n}\mathcal{N}\left(x_{kj};\overline{x}_{kj},\widetilde{x}_{kj}\right).$$

The VB cost function is then the sum of the following terms:

$$C_{\mathrm{vb}} = \sum_{ij\in O}C_{yij} + \sum_{i=1}^{d}C_{mi} + \sum_{i=1}^{d}\sum_{k=1}^{c}C_{wik} + \sum_{k=1}^{c}\sum_{j=1}^{n}C_{xkj},$$

where the individual terms are

$$C_{yij} = \frac{1}{2v_y}\left[(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}}\overline{\mathbf{x}}_j - \overline{m}_i)^2 + \widetilde{m}_i + \sum_{k=1}^{c}\left(\widetilde{w}_{ik}\overline{x}_{kj}^2 + \overline{w}_{ik}^2\widetilde{x}_{kj} + \widetilde{w}_{ik}\widetilde{x}_{kj}\right)\right] + \frac{1}{2}\log 2\pi v_y,$$

$$C_{mi} = \left\langle\log\frac{q(m_i)}{p(m_i)}\right\rangle = \frac{\overline{m}_i^2 + \widetilde{m}_i}{2v_m} - \frac{1}{2}\log\frac{\widetilde{m}_i}{v_m} - \frac{1}{2},$$

$$C_{wik} = \left\langle\log\frac{q(w_{ik})}{p(w_{ik})}\right\rangle = \frac{\overline{w}_{ik}^2 + \widetilde{w}_{ik}}{2v_{w,k}} - \frac{1}{2}\log\frac{\widetilde{w}_{ik}}{v_{w,k}} - \frac{1}{2},$$

$$C_{xkj} = \left\langle\log\frac{q(x_{kj})}{p(x_{kj})}\right\rangle = \frac{1}{2}(\overline{x}_{kj}^2 + \widetilde{x}_{kj}) - \frac{1}{2}\log\widetilde{x}_{kj} - \frac{1}{2}.$$

Variational parameters $\overline{m}_i$, $\widetilde{m}_i$, $v_{w,k}$ and v_m are updated to minimize the cost function using the update rules in (49)–(51). Parameters $\widetilde{w}, \widetilde{x}$ can be updated to minimize the cost function:

$$\widetilde{w}_{ik} = v_y \left[\frac{v_y}{v_{w,k}} + \sum_{j \in O_i} \left(\overline{x}_{kj}^2 + \widetilde{x}_{kj} \right) \right]^{-1}, \tag{52}$$

$$\widetilde{x}_{kj} = v_y \left[v_y + \sum_{i \in O_j} \left(\overline{w}_{ik}^2 + \widetilde{w}_{ik} \right) \right]^{-1}. \tag{53}$$

The update rule for v_y is

$$v_y = \frac{1}{N} \sum_{ij \in O} \left[(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}} \overline{\mathbf{x}}_j - \overline{m}_i)^2 + \widetilde{m}_i + \sum_{k=1}^{c} \left(\widetilde{w}_{ik} \overline{x}_{kj}^2 + \overline{w}_{ik}^2 \widetilde{x}_{kj} + \widetilde{w}_{ik} \widetilde{x}_{kj} \right) \right].$$

Minimizing the cost function w.r.t. each parameter $\overline{w}_{ij}$ keeping the others fixed would lead to a slow algorithm (e.g., Honkela et al., 2003). Instead, derivatives required for gradient-based optimization are:

$$\frac{\partial C_{\mathrm{vb}}}{\partial \overline{w}_{ik}} = \frac{\overline{w}_{ik}}{v_{w,k}} + \frac{1}{v_y} \sum_{j \in O_i} \left[-\left(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}} \overline{\mathbf{x}}_j - \overline{m}_i \right) \overline{x}_{kj} + \overline{w}_{ik} \widetilde{x}_{kj} \right],$$

$$\frac{\partial C_{\mathrm{vb}}}{\partial \overline{x}_{kj}} = \overline{x}_{kj} + \frac{1}{v_y} \sum_{i \in O_j} \left[-\left(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}} \overline{\mathbf{x}}_j - \overline{m}_i \right) \overline{w}_{ik} + \widetilde{w}_{ik} \overline{x}_{kj} \right].$$

The second-order derivatives which can be used to speed up learning, as explained in Section 6.1, coincide with the inverse of the updated variances given in (52)–(53): $\partial^2 C_{\mathrm{vb}} / \partial \overline{w}_{ik}^2 = \widetilde{w}_{ik}^{-1}$ and $\partial^2 C_{\mathrm{vb}} / \partial \overline{x}_{kj}^2 = \widetilde{x}_{kj}^{-1}$.

Appendix E. MAPPCA

The cost function C_{MAP} is given in (39). The alternating optimization procedure can be implemented as follows:

$$\mathbf{x}_j = \left(v_y \mathbf{I} + \sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}} \right)^{-1} \sum_{i \in O_j} \mathbf{w}_i (y_{ij} - m_i), \qquad j = 1, \ldots, n,$$

$$m_i = \frac{v_m}{|O_i|(v_m + v_y/|O_i|)} \sum_{j \in O_i} \left[y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j \right],$$

$$\mathbf{w}_i = \left(v_y \, \mathrm{diag}(v_{w,k}^{-1}) + \sum_{j \in O_i} \mathbf{x}_j \mathbf{x}_j^{\mathrm{T}} \right)^{-1} \sum_{j \in O_i} \mathbf{x}_j (y_{ij} - m_i), \qquad i = 1, \ldots, d,$$

$$v_y = \frac{1}{N} \sum_{ij \in O} (y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i)^2,$$

$$v_{w,k} = \frac{1}{d} \sum_{i=1}^{d} w_{ik}^2, \qquad v_m = \frac{1}{d} \sum_{i=1}^{d} m_i^2.$$

Gradient-based learning discussed in Section 6.1 requires the following derivatives:

$$\frac{\partial C_{\mathrm{MAP}}}{\partial w_{ik}} = \frac{w_{ik}}{v_{w,k}} + \frac{1}{v_y} \sum_{j \in O_i} \left[-\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i\right) x_{kj} \right],$$

$$\frac{\partial C_{\mathrm{MAP}}}{\partial x_{kj}} = x_{kj} + \frac{1}{v_y} \sum_{i \in O_j} \left[-\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i\right) w_{ik} \right],$$

$$\frac{\partial^2 C_{\mathrm{MAP}}}{\partial w_{ik}^2} = \frac{1}{v_{w,k}} + \frac{1}{v_y} \sum_{j \in O_i} x_{kj}^2,$$

$$\frac{\partial^2 C_{\mathrm{MAP}}}{\partial x_{kj}^2} = 1 + \frac{1}{v_y} \sum_{i \in O_j} w_{ik}^2 .$$

References

T. W. Anderson. Maximum likelihood estimates for a multivariate normal distribution when some observations are missing. *Journal of the American Statistical Association*, 52:200–203, 1957.

R. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2007)*, 2007.

R. Bell, Y. Koren, and C. Volinsky. The BellKor solution to the Netflix Prize. Available at `http://www.netflixprize.com/`, 2007.

C. M. Bishop. Variational principal components. In *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN99)*, pages 509–514, 1999.

C. M. Bishop. *Pattern Recognition and Machince Learning*. Springer, Cambridge, 2006.

W. J. Boscardin and X. Zhang. Modeling the covariance and correlation matrix of repeated measures. In A. Gelman and X.-L. Meng, editors, *Applied Bayesian Modeling and Causal Inference from an Incomplete-Data Perspective*. John Wiley & Sons, New York, 2004.

A. Christoffersson. *The One Component Model with Incomplete Data*. PhD thesis, Uppsala University, 1970.

A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing - Learning Algorithms and Applications*. Wiley, 2002.

C. L. de Ligny, G. H. E. Nieuwdorp, W. K. Brederode, W. E. Hammers, and J. C. van Houwelingen. An application of factor analysis with missing data. *Technometrics*, 23(1):91–95, 1981.

R. E. Dear. A principal components missing data method for multiple regression models. Technical Report SP-86, Santa Monica: Systems Development Corporation, 1959.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

K. Diamantaras and S. Kung. *Principal Component Neural Networks - Theory and Application*. Wiley, 1996.

S. Funk. Netflix update: Try this at home. Available at `http://sifter.org/~simon/journal/20061211.html`, December 2006.

Z. Ghahramani and M. I. Jordan. Learning from incomplete data. Technical report CBCL-108, Massachusetts Institute of Technology, 1994.

B. Grung and R. Manne. Missing values in principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 42(1):125–139, 1998.

S. Haykin. *Modern Filters*. Macmillan, 1989.

G. E. Hinton and D. van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory*, pages 5–13, Santa Cruz, CA, USA, 1993.

P. Hoff. Model averaging and dimension selection for the singular value decomposition. *Journal of the American Statistical Association*, 102(478):674–685, 2008.

T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

A. Honkela, H. Valpola, and J. Karhunen. Accelerating cyclic update algorithms for parameter estimation by pattern searches. *Neural Processing Letters*, 17(2):191–203, 2003.

A. Ilin and H. Valpola. On the effect of the form of the posterior approximation in variational learning of ICA models. *Neural Processing Letters*, 22(2):183–204, 2005.

I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2nd edition, 2002.

A. Kaplan, Y. Kushnir, M. Cane, and M. Blumenthal. Reduced space optimal analysis for historical datasets: 136 years of Atlantic sea surface temperatures. *Journal of Geophysical Research*, 102: 27835–27860, 1997.

Y. J. Lim and Y. W. Teh. Variational Bayesian approach to movie rating prediction. In *Proceedings of of KDD Cup and Workshop 2007, held during the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, California, 2007.

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. J. Wiley & Sons, 1987.

J. Luttinen and A. Ilin. Transformations in variational Bayesian factor analysis to speed up learning. *Neurocomputing*, 73(7–9):1093–1102, 2010.

T. Minka. Automatic choice of dimensionality for PCA. In V. Tresp T. Leen, T. Dietterich, editor, *Advances in Neural Information Processing Systems 13*, pages 598–604. MIT Press, Cambridge, MA, USA, 2001.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. The MIT Press, Cambridge, MA, USA, 1999.

Netflix. Netflix prize webpage, 2007. http://www.netflixprize.com/.

S. Oba, M. Sato, I. Takemasa, M. Monden, K. Matsubara, and S. Ishii. A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, 2003.

E. Oja. *Subspace Methods of Pattern Recognition*. Research Studies Press and J. Wiley, 1983.

A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, 2007.

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

T. Raiko and H. Valpola. Missing values in nonlinear factor analysis. In *Proceedings of the 8th International Conference on Neural Information Processing (ICONIP'01)*, pages 822–827, Shanghai, 2001.

T. Raiko, A. Ilin, and J. Karhunen. Principal component analysis for large scale problems with lots of missing values. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, pages 691–698, Warsaw, Poland, 2007a.

T. Raiko, H. Valpola, M. Harva, and J. Karhunen. Building blocks for variational Bayesian learning of latent variable models. *Journal of Machine Learning Research*, 8:155–201, 2007b.

T. Raiko, A. Ilin, and J. Karhunen. Principal component analysis for sparse high-dimensional data. In *Proceedings of the 14th International Conference on Neural Information Processing (ICONIP 2007)*, pages 566–575, Kitakyushu, Japan, 2008.

S. Roweis. EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems 10*, pages 626–632, Cambridge, MA, 1998. MIT Press.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning (ICML-2008)*, Helsinki, Finland, 2008a.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20*, 2008b.

R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning (ICML-2007)*, 2007.

N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*, Washington DC, 2003.

The Ensemble. Website of the runner-up in the Netflix competition, 2009. http://www.the-ensemble.com/.

M. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

C. S. Wallace. Classification by minimum-message-length inference. In S. G. Aki, F. Fiala, and W. W. Koczkodaj, editors, *Advances in Computing and Information – ICCI '90*, volume 468 of *Lecture Notes in Computer Science*, pages 72–81. Springer, Berlin, 1990.

T. Wiberg. Computation of principal components when data are missing. In Johannes Gordesch and Peter Naeve, editors, *COMPSTAT 1976: Proceedings in Computational Statistics, 2nd symposium held in Berlin (West)*, pages 229–236. Wien: Physica-Verlag, 1976.

G. Young. Maximum likelihood estimation and factor analysis. *Psychometrika*, 6(1):49–53, 1941.

Journal of Machine Learning Research 11 (2010) 2001-2049 Submitted 12/09; Published 6/10

Posterior Regularization for Structured Latent Variable Models

Kuzman Ganchev KUZMAN@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Levine 302, 3330 Walnut St
Philadelphia PA, 19104, USA

João Graça JOAO.GRACA@L2F.INESC-ID.PT
L2F Inesc-ID Spoken Language Systems Lab
R. Alves Redol, 9
1000-029 Lisboa, Portugal

Jennifer Gillenwater JENGI@CIS.UPENN.EDU
Ben Taskar TASKAR@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Levine 302, 3330 Walnut St
Philadelphia PA, 19104, USA

Editor: Michael Collins

Abstract

We present posterior regularization, a probabilistic framework for structured, weakly supervised learning. Our framework efficiently incorporates indirect supervision via constraints on posterior distributions of probabilistic models with latent variables. Posterior regularization *separates* model complexity from the complexity of structural constraints it is desired to satisfy. By directly imposing decomposable regularization on the posterior moments of latent variables during learning, we retain the computational efficiency of the unconstrained model while ensuring desired constraints hold in expectation. We present an efficient algorithm for learning with posterior regularization and illustrate its versatility on a diverse set of structural constraints such as bijectivity, symmetry and group sparsity in several large scale experiments, including multi-view learning, cross-lingual dependency grammar induction, unsupervised part-of-speech induction, and bitext word alignment.[1]

Keywords: posterior regularization framework, unsupervised learning, latent variables models, prior knowledge, natural language processing

1. Introduction

In unsupervised problems where data has sequential, recursive, spatial, relational, and other kinds of structure, we often employ structured statistical models with latent variables to tease apart the underlying dependencies and induce meaningful semantic categories. Unsupervised part-of-speech and grammar induction, and word and phrase alignment for statistical machine translation in natural language processing are examples of such aims. Generative models (probabilistic grammars,

1. A preliminary version of the PR framework appeared in Graça et al. (2007). Various extensions and applications appeared in Ganchev et al. (2008a), Ganchev et al. (2008b), Ganchev et al. (2009), Graça et al. (2009a) and Graça et al. (2010).

graphical models, etc.) are usually estimated by maximizing the likelihood of the observed data by marginalizing over the hidden variables, typically via the Expectation Maximization (EM) algorithm. Because of computational and statistical concerns, generative models used in practice are very simplistic models of the underlying phenomena; for example, the syntactic structure of language or the language translation process. A pernicious problem with such models is that marginal likelihood may not guide the model towards the intended role for the latent variables, instead focusing on explaining irrelevant but common correlations in the data. Since we are mostly interested in the distribution of the latent variables in the hope that they capture *intended* regularities without direct supervision, controlling this latent distribution is critical. Less direct methods such as clever initialization, ad hoc procedural modifications, and complex data transformations are often used to affect the posteriors of latent variables in a desired manner.

A key challenge for structured, weakly supervised learning is developing a flexible, declarative framework for expressing structural constraints on latent variables arising from prior knowledge and indirect supervision. Structured models have the ability to capture a very rich array of possible relationships, but adding complexity to the model often leads to intractable inference. In this article, we present the posterior regularization (PR) framework (Graça et al., 2007), which *separates* model complexity from the complexity of structural constraints it is desired to satisfy. Unlike parametric regularization in a Bayesian framework, our approach incorporates data-dependent constraints that are easy to encode as information about model *posteriors* on the observed data, but may be difficult to encode as information about model parameters through Bayesian *priors*. In Sections 5-8 we describe a variety of such useful prior knowledge constraints in several application domains.

The contributions of this paper are:

- A flexible, declarative framework for structured, weakly supervised learning via posterior regularization.
- An efficient algorithm for model estimation with posterior regularization.
- An extensive evaluation of different types of constraints in several domains: multi-view learning, cross-lingual dependency grammar induction, unsupervised part-of-speech induction, and bitext word alignment.
- A detailed explanation of the connections between several other recent proposals for weak supervision, including structured constraint-driven learning (Chang et al., 2007), generalized expectation criteria (Mann and McCallum, 2008, 2007) and Bayesian measurements (Liang et al., 2009).

The rest of this paper is organized as follows. Section 2 describes the posterior regularization framework and Section 3 illustrates the range of different types of weak supervision constraints representable in our framework. Section 4 describes the relationship between posterior regularization and other related frameworks. Sections 5-8 describe applications of PR to several problems: word alignment (§5), multi-view learning (§6), cross-lingual projection (§7) and inducing sparsity structure (§8). Section 9 concludes the paper and presents areas for future work.

2. Posterior Regularization Framework

In this section we describe the posterior regularization framework, which incorporates side-information into parameter estimation in the form of linear constraints on posterior expectations. As we will

show, this allows tractable learning and inference even when the constraints would be intractable to encode directly in the model parameters. By defining a flexible language for specifying diverse types of problem-specific prior knowledge, we make the framework applicable to a wide variety of probabilistic models, both generative and discriminative. In Sections 2.1-2.7 we will focus on generative models, and describe the case of discriminative models in Section 2.8. We will use a problem from natural language processing as a running example in the exposition:

> **Running Example** *The task is part-of-speech (POS) tagging with limited or no training data. Suppose we know that each sentence should have at least one verb and at least one noun, and would like our model to capture this constraint on the unlabeled sentences. The model we will be using is a first-order hidden Markov model (HMM).*

We describe four other applications with empirical results in Sections 5-8, but it will be easier to illustrate key concepts using this simple example.

2.1 Preliminaries and Notation

We assume that there is a natural division of variables into "input" variables $\mathbf{x}$ and "target" variables $\mathbf{y}$ for each data instance, where $\mathbf{x}$'s are always observed. We denote the set of all instances of unlabeled data as $\mathbf{X}$. In case of semi-supervised learning, we have some labeled data as well, and we will use the notation $(\mathbf{X}_L, \mathbf{Y}_L)$ to denote all the labeled instances.

The starting point for using the PR framework is a probabilistic model. Let θ be the parameters of the model. For now we assume a generative model $p_\theta(\mathbf{x}, \mathbf{y})$, and we use $\mathcal{L}(\theta) = \log p_\theta(\mathbf{X}_L, \mathbf{Y}_L) + \log \sum_{\mathbf{Y}} p_\theta(\mathbf{X}, \mathbf{Y}) + \log p(\theta)$ to denote the parameter-regularized log-likelihood of the data.

> **Running Example** *In the POS tagging example from above, we would use* $\mathbf{x} = \{x_1, x_2, \ldots x_{|\mathbf{x}|}\}$ *to denote a sentence (i.e., a sequence of words* x_i*) and* $\mathbf{y} = \{y_1, y_2, \ldots y_{|\mathbf{x}|}\}$ *to denote a possible POS assignment. Using an HMM, it is defined in the normal way as:*
>
> $$p_\theta(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{|\mathbf{x}|} p_\theta(y_i|y_{i-1}) \; p_\theta(x_i|y_i),$$
>
> *with* θ *representing the multinomial distributions directly, and where* $p_\theta(y_1|y_0) = p_\theta(y_1)$ *represents a set of initial probabilities. Suppose we have a small labeled corpus and a larger unlabeled corpus. For a generative model such as an HMM, the log-likelihood (+ log-prior) is:*
>
> $$\mathcal{L}(\theta) = \log p_\theta(\mathbf{X}_L, \mathbf{Y}_L) + \log \sum_{\mathbf{Y}} p_\theta(\mathbf{X}, \mathbf{Y}) + \log p(\theta),$$
>
> *where corpus probabilities are products over instances:* $p_\theta(\mathbf{x}, \mathbf{y}) = \prod p_\theta(\mathbf{x}, \mathbf{y})$ *and analogously for* $\mathbf{X}_L, \mathbf{Y}_L$*; and where* $p(\theta)$ *is a prior distribution over the parameters* θ.

2.2 Regularization via Posterior Constraints

The goal of the posterior regularization framework is to restrict the space of the model posteriors on unlabeled data as a way to guide the model towards desired behavior. In this section we describe a version of PR specified with respect to a set of constraints. In this case, posterior information is

specified with sets $\mathcal{Q}$ of allowed distributions over the hidden variables $\mathbf{y}$. We will define $\mathcal{Q}$ in terms of *constraint* features $\phi(\mathbf{X}, \mathbf{Y})$ and their expectations.[2]

Running Example *Recall that in our running example, we want to bias learning so that each sentence is labeled to contain at least one verb. To encode this formally, we define a feature* $\phi(\mathbf{x}, \mathbf{y}) =$ *"number of verbs in* $\mathbf{y}$*", and require that this feature has expectation at least 1. For consistency with the rest of the exposition and standard optimization literature, we will use the equivalent* $\phi(\mathbf{x}, \mathbf{y}) =$ *"negative number of verbs in* $\mathbf{y}$*" and require this has expectation at most -1.*[3]

$$\mathcal{Q}_{\mathbf{x}} = \{q_{\mathbf{x}}(\mathbf{y}) \ : \ \mathbf{E}_q[\phi(\mathbf{x}, \mathbf{y})] \leq -1\}.$$

Note that we enforce the constraint only in expectation, so there might be a labeling with non-zero probability that does not contain a verb. To actually enforce this constraint in the model would break the first-order Markov property of the distribution.[4] *In order to also require at least one noun per sentence in expectation, we would add another constraint feature, so that* ϕ *would be a function from* $\mathbf{x}, \mathbf{y}$ *pairs to* $\mathbb{R}^2$.

We define $\mathcal{Q}$, the set of valid distributions, with respect to the *expectations* of constraint features, rather than their probabilities, so that our objective leads to an efficient algorithm. As we will see later in this section, we also require that the constraint features decompose as a sum in order to ensure an efficient algorithm. More generally than in the running example, we will define constraints over an entire corpus:

$$\textbf{Constrained Posterior Set}: \qquad \mathcal{Q} = \{q(\mathbf{Y}) : \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] \leq \mathbf{b}\}.$$

In words, $\mathcal{Q}$ denotes the region where constraint feature expectations are bounded by $\mathbf{b}$. Additionally, it is often useful to allow small violations whose norm is bounded by $\varepsilon \geq 0$:

$$\textbf{Constrained Set (with slack)}: \quad \mathcal{Q} = \{q(\mathbf{Y}) : \exists \xi, \ \mathbf{E}_q[\phi(\mathbf{X}, \mathbf{Y})] - \mathbf{b} \leq \xi; \ ||\xi||_\beta \leq \varepsilon\}. \qquad (1)$$

Here ξ is a vector of slack variables and $||\cdot||_\beta$ denotes some norm. Note that the PR method we describe will only be useful if $\mathcal{Q}$ is non-empty:

Assumption 2.1 *$\mathcal{Q}$ is non-empty.*

We explore several types of constraints in Sections 5-8, including: constraints similar to the running example, where each hidden state is constrained to appear at most once in expectation; constraints that bias two models to agree on latent variables in expectation; constraints that enforce a particular group-sparsity of the posterior moments. The constraint set defined in Equation 1 is usually referred to as inequality constraints with slack, since setting $\varepsilon = 0$ enforces inequality constraints strictly. The derivations for equality constraints are very similar to the derivations for inequality so we leave them out in the interest of space. Note also that we can encode equality

2. Note: the constraint features do not appear anywhere in the model. If the model has a log-linear form, then it would be defined with respect to a different set of *model* features, not related to the *constraint* features we consider here.
3. Note that the distribution $q_{\mathbf{x}}$ and $\mathcal{Q}_{\mathbf{x}}$ depend on $\mathbf{x}$ because the features $\phi(\mathbf{x}, \mathbf{y})$ might depend on the particular example $\mathbf{x}$.
4. At every position in the sentence, we would need to know whether a verb was used at any other position.

Symbol	Meaning
$\mathbf{x}$	(observed) input variables for a particular example
$\mathbf{y}$	(usually hidden) output variables for a particular example
$\mathbf{X}, \mathbf{Y}$	$\mathbf{x}$ and $\mathbf{y}$ for the entire unlabeled portion of the corpus
$\mathbf{X}_L, \mathbf{Y}_L$	$\mathbf{x}$ and $\mathbf{y}$ for the entire labeled portion of the corpus (possibly empty)
$p_\theta(\mathbf{x}, \mathbf{y})$	a generative, joint model with parameters θ
$\mathcal{L}(\theta)$	data log-likelihood and parameter prior: $\log p_\theta(\mathbf{X}_L, \mathbf{Y}_L) + \log \sum_{\mathbf{Y}} p_\theta(\mathbf{X}, \mathbf{Y}) + \log p(\theta)$
$\mathcal{Q}_\mathbf{x}, \mathcal{Q}$	posterior regularization set: constrained set of desired data-conditional distributions
$\phi(\mathbf{x}, \mathbf{y})$	constraint features: used to encode posterior regularization
$\mathbf{b}$	bounds on the desired expected values of constraint features
ξ	slack variables used to allow small violations of constraints
$J_\mathcal{Q}(\theta)$	posterior regularized likelihood: $\mathcal{L}(\theta) - \mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}\|\mathbf{X}))$

Table 1: Summary of notation used.

constraints by adding two inequality constraints, although this will leave us with twice as many variables in the dual. The assumption of linearity of the constraints is computationally important, as we will show below. For now, we do not make any assumptions about the features $\phi(\mathbf{x}, \mathbf{y})$, but if they factor in the same way as the model, then we can use the same inference algorithms in PR training as we use for the original model (see Proposition 2.2). In PR, the log-likelihood of a model is penalized with the KL-divergence between the desired distribution space $\mathcal{Q}$ and the model posteriors,

$$\mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X})) = \min_{q \in \mathcal{Q}} \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})).$$

The posterior-regularized objective is:

$$\textbf{Posterior Regularized Likelihood}: \quad J_\mathcal{Q}(\theta) = \mathcal{L}(\theta) - \mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X})). \tag{2}$$

The objective trades off likelihood and distance to the desired posterior subspace (modulo getting stuck in local maxima) and provides an effective means of controlling the posteriors. In many cases, prior knowledge is easy to specify in terms of posteriors, and much more difficult to specify as priors on model parameters or by explicitly adding constraints to the model. A key advantage of using regularization on posteriors is that the learned model itself remains simple and tractable, while during learning it is driven to obey the constraints through setting appropriate parameters θ. The advantage of imposing the constraints via KL-divergence from the posteriors is that the objective above can be optimized using a simple EM scheme described in Section 2.6. It is also possible to use a similar algorithm to maximize $\mathcal{L}(\theta) - \alpha\mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y} \mid \mathbf{X}))$, for $\alpha \in [0,1]$. See Appendix A for details. Note that the algorithm we will present in Section 2.6 will not allow us to optimize an objective with $\alpha > 1$, and this leads us to have both a KL-penalty term in Equation 2 and also to potentially have slack in the definition of the constraint set $\mathcal{Q}$. We do not need to allow slack in the objective, as long as we are sure that the constraint set $\mathcal{Q}$ is non-empty. At increased computational cost, it is also possible to eliminate the KL-penalty portion of the objective, instead directly constraining the model's posterior distribution to be inside the constraint set $p_\theta(\mathbf{Y}|\mathbf{X}) \in \mathcal{Q}$. See Section 4 for details. Figure 1 illustrates the objective in Equation 2. Normal maximum likelihood training is equivalent to minimizing the KL distance between the distribution concentrated on $\mathbf{X}$ and the set of distributions representable by the model. Any particular setting of the model parameters results in

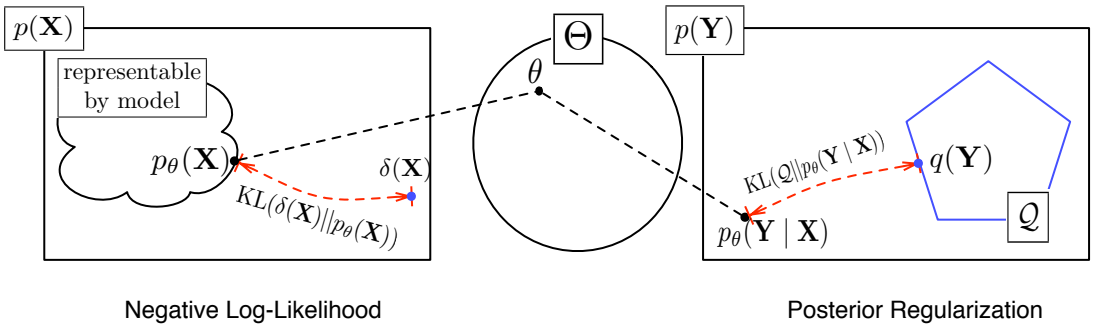


Figure 1: An illustration of the PR objective for generative models, as a sum of two KL terms. The symbol Θ represents the set of possible model parameters, $\delta(\mathbf{X})$ is a distribution that puts probability 1 on $\mathbf{X}$ and 0 on all other assignments. Consequently $\mathbf{KL}(\delta(\mathbf{X})||p_\theta(\mathbf{X})) = \mathcal{L}(\theta)$. (We ignore the parameter prior and additional labeled data in this figure for clarity.)

the posterior distribution $p_\theta(\mathbf{Y}|\mathbf{X})$. PR adds to the maximum likelihood objective a corresponding KL distance for this distribution. If Q has only one distribution, then we recover labeled maximum likelihood training. This is one of the justifications for the use and the particular direction of the KL distance in the penalty term.

> **Running Example** *In order to represent a corpus-wide constraint set Q for our POS problem, we stack the constraint features into a function from $\mathbf{X},\mathbf{Y}$ pairs (sentences, part-of-speech sequences) to $\mathbb{R}^{2|\mathbf{X}|}$, where $|\mathbf{X}|$ is the number of sentences in our unlabeled corpus. For the POS tagging example, the PR objective penalizes parameters that do not assign each sentence at least one verb and one noun in expectation.*

For PR to be successful, the model $p_\theta(\mathbf{Y}|\mathbf{X})$ has to be expressive enough to ensure that the learned model has posteriors $p_\theta(\mathbf{Y}|\mathbf{X})$ in or nearly in Q. Even if that is the case, the same parameters might not ensure that the constraints are satisfied on a test corpus, so we could also use $q(\mathbf{Y}) = \arg\min_{q' \in Q} \mathbf{KL}(q'(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X}))$ for prediction instead of $p_\theta(\mathbf{Y}|\mathbf{X})$. We will see in Sections 5 and 7 that this sometimes results in improved performance. Chang et al. (2007) report similar results for their constraint-driven learning framework.

2.3 Slack Constraints vs. Penalty

In order for our objective to be well defined, Q must be non-empty. When there are a large number of constraints, or when the constraint features ϕ are defined by some instance-specific process, it might not be easy to choose constraint values $\mathbf{b}$ and slack ε that lead to satisfiable constraints. It is sometimes easier to penalize slack variables instead of setting a bound ε on their norm. In these cases, we add a slack penalty to the regularized likelihood objective in Equation 2:

$$\begin{aligned} \mathcal{L}(\theta) \quad - \min_{q,\xi} \quad & \mathbf{KL}\left(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})\right) + \sigma||\xi||_\beta \\ & \text{s.t.} \quad \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi. \end{aligned} \tag{3}$$

The slack-constrained and slack-penalized versions of the objectives are equivalent in the sense that they follow the same regularization path: for every ε there exists some σ that results in identical parameters θ. Note that while we have used a norm $||\cdot||_\beta$ to impose a cost on violations of the constraints, we could have used any arbitrary convex penalty function, for which the minimal q is easily computable.

2.4 Computing the Posterior Regularizer

In this subsection, we describe how to compute the objective we have introduced for fixed parameters θ. The regularization term is stated in Equations 2 and 3 in terms of an optimization problem. We assume that we have algorithms to do inference[5] in the statistical model of interest, p_θ. We describe the computation of the regularization term for the inequality constraints:

$$\min_{q,\xi} \quad \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \qquad \text{s.t.} \quad \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi; \quad ||\xi||_\beta \leq \varepsilon. \tag{4}$$

Proposition 2.1 *The regularization problems for PR with inequality constraints in Equation 4 can be solved efficiently in its dual form. The primal solution q^* is unique since KL divergence is strictly convex and is given in terms of the dual solution λ^* by:*

$$q^*(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{X}) \exp\{-\lambda^* \cdot \phi(\mathbf{X},\mathbf{Y})\}}{Z(\lambda^*)} \tag{5}$$

where $Z(\lambda^*) = \sum_{\mathbf{Y}} p_\theta(\mathbf{Y}|\mathbf{X}) \exp\{-\lambda^* \cdot \phi(\mathbf{X},\mathbf{Y})\}$. Define $||\cdot||_{\beta^*}$ as the dual norm of $||\cdot||_\beta$. The dual of the problem in Equation 4 is:

$$\max_{\lambda \geq 0} \quad -\mathbf{b} \cdot \lambda - \log Z(\lambda) \; - \varepsilon \, ||\lambda||_{\beta^*}. \tag{6}$$

The proof is included in Appendix B using standard Lagrangian duality results and strict convexity of KL (e.g., Bertsekas, 1999). The dual form in Equation 6 is typically computationally more tractable than the primal form (Equation 4) because there is one dual variable per expectation constraint, while there is one primal variable per labeling $\mathbf{Y}$. For structured models, this is typically intractable. An analogous proposition can be proven for the objective with penalties (Equation 3), with almost identical proof. We omit this for brevity.

2.5 Factored $q(\mathbf{Y})$ for Factored Constraints

The form of the optimal q with respect to $p_\theta(\mathbf{Y}|\mathbf{X})$ and ϕ has important computational implications.

Proposition 2.2 *If $p_\theta(\mathbf{Y}|\mathbf{X})$ factors as a product of clique potentials over a set of cliques $\mathcal{C}$, and $\phi(\mathbf{X},\mathbf{Y})$ factors as a sum over some subset of those cliques, then the optimizer $q^*(\mathbf{Y})$ of Equation 5 will also factor as a product of potentials of cliques in $\mathcal{C}$.*

This is easy to show. Our assumptions are a factorization for p_θ:

$$\textbf{Factored Posteriors}: \quad p(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_{c \in \mathcal{C}} \psi(\mathbf{X}, \mathbf{Y}_c)$$

5. Specifically, we need to be able to compute marginal distributions efficiently.

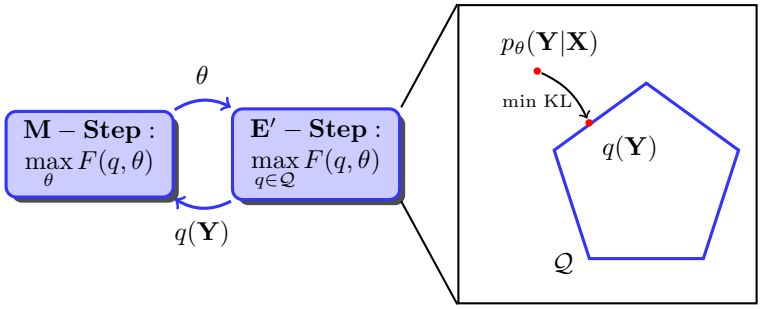


Figure 2: Modified EM for optimizing generative PR objective $\mathcal{L}(\theta) - \mathbf{KL}(Q \,\|\, p_\theta(\mathbf{Y}|\mathbf{X}))$.

and the same factorization for ϕ:

$$\textbf{Factored Features}: \quad \phi(\mathbf{X},\mathbf{Y}) = \sum_{c\in\mathcal{C}} \phi(\mathbf{X},\mathbf{Y}_c)$$

which imply that $q^*(\mathbf{Y})$ will also factor as a product over the cliques $\mathcal{C}$:

$$\begin{aligned} \textbf{Factored Solution}: \quad q^*(\mathbf{Y}) &= \frac{1}{Z(\mathbf{X})Z(\lambda)} \prod_{c\in\mathcal{C}} \psi(\mathbf{X},\mathbf{Y}_c)\, \exp\{-\lambda\cdot\phi(\mathbf{X},\mathbf{Y}_c)\} \\ &= \frac{1}{Z'(\mathbf{X})} \prod_{c\in\mathcal{C}} \psi'(\mathbf{X},\mathbf{Y}_c), \end{aligned}$$

where $\psi'(\mathbf{X},\mathbf{Y}_c) = \psi(\mathbf{X},\mathbf{Y}_c)\exp\{-\lambda\cdot\phi(\mathbf{X},\mathbf{Y}_c)\}$ and $Z'(\mathbf{X}) = Z(\mathbf{X})Z(\lambda)$.

2.6 Generative Posterior Regularization via Expectation Maximization

This section presents an optimization algorithm for the PR objective. The algorithm we present is a minorization-maximization algorithm akin to EM, and both slack-constrained and slack-penalized formulations can be optimized using it. To simplify the exposition, we focus first on slack-constrained version, and leave a treatment of optimization of the slack-penalized version to Section 2.7.

Recall the standard expectation maximization (EM) algorithm used to optimize marginal likelihood $\mathcal{L}(\theta) = \log \sum_{\mathbf{Y}} p_\theta(\mathbf{X},\mathbf{Y})$. Again, for clarity of exposition, we ignore $\log p(\theta)$, the prior on θ, as well as $\log p_\theta(\mathbf{X}_L,\mathbf{Y}_L)$, the labeled data term, as they are simple to incorporate, just as in regular EM. Neal and Hinton (1998) describe an interpretation of the EM algorithm as block coordinate ascent on a function that lower-bounds $\mathcal{L}(\theta)$, which we also use below. By Jensen's inequality, we define a lower-bound $F(q,\theta)$ as

$$\mathcal{L}(\theta) = \log \sum_{\mathbf{Y}} q(\mathbf{Y}) \frac{p_\theta(\mathbf{X},\mathbf{Y})}{q(\mathbf{Y})} \geq \sum_{\mathbf{Y}} q(\mathbf{Y}) \log \frac{p_\theta(\mathbf{X},\mathbf{Y})}{q(\mathbf{Y})} = F(q,\theta).$$

we can re-write $F(q,\theta)$ as

$$\begin{aligned} F(q,\theta) &= \sum_{\mathbf{Y}} q(\mathbf{Y}) \log(p_\theta(\mathbf{X}) p_\theta(\mathbf{Y}|\mathbf{X})) - \sum_{\mathbf{Y}} q(\mathbf{Y}) \log q(\mathbf{Y}) \\ &= \mathcal{L}(\theta) - \sum_{\mathbf{Y}} q(\mathbf{Y}) \log \frac{q(\mathbf{Y})}{p_\theta(\mathbf{Y}|\mathbf{X})} \\ &= \mathcal{L}(\theta) - \mathbf{KL}(q(\mathbf{Y}) || p_\theta(\mathbf{Y}|\mathbf{X})). \end{aligned}$$

Using this interpretation, we can view EM as performing coordinate ascent on $F(q,\theta)$. Starting from an initial parameter estimate θ^0, the algorithm iterates two block-coordinate ascent steps until a convergence criterion is reached:

$$\begin{aligned} \mathbf{E}: q^{t+1} &= \arg\max_q F(q,\theta^t) = \arg\min_q \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y} \mid \mathbf{X})), \\ \mathbf{M}: \theta^{t+1} &= \arg\max_\theta F(q^{t+1},\theta) = \arg\max_\theta \mathbf{E}_{q^{t+1}}[\log p_\theta(\mathbf{X},\mathbf{Y})]. \end{aligned} \tag{7}$$

It is easy to see that the E-step sets $q^{t+1}(\mathbf{Y}) = p_{\theta^t}(\mathbf{Y}|\mathbf{X})$.

The PR objective (Equation 2) is

$$J_\mathcal{Q}(\theta) = \max_{q \in \mathcal{Q}} F(q,\theta) = \mathcal{L}(\theta) - \min_{q(\mathbf{Y}) \in \mathcal{Q}} \mathbf{KL}(q(\mathbf{Y}) || p_\theta(\mathbf{Y}|\mathbf{X})),$$

where $\mathcal{Q} = \{q(\mathbf{Y}) : \exists \xi,\ \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi;\ ||\xi||_\beta \leq \varepsilon\}$. In order to optimize this objective, it suffices to modify the E-step to include the constraints:

$$\mathbf{E}': q^{t+1} = \arg\max_{q \in \mathcal{Q}} F(q,\theta^t) = \arg\min_{q \in \mathcal{Q}} \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y}|\mathbf{X})). \tag{8}$$

The projected posteriors $q^{t+1}(\mathbf{Y})$ are then used to compute sufficient statistics and update the model's parameters in the M-step, which remains unchanged, as in Equation 7. This scheme is illustrated in Figure 2.

Proposition 2.3 *The modified EM algorithm illustrated in Figure 2, which iterates the modified E-step (Equation 8) with the normal M-step (Equation 7), monotonically increases the PR objective:* $J_\mathcal{Q}(\theta^{t+1}) \geq J_\mathcal{Q}(\theta^t)$.

Proof: The proof is analogous to the proof of monotonic increase of the standard EM objective. Essentially,

$$J_\mathcal{Q}(\theta^{t+1}) = F(q^{t+2},\theta^{t+1}) \geq F(q^{t+1},\theta^{t+1}) \geq F(q^{t+1},\theta^t) = J_\mathcal{Q}(\theta^t).$$

The two inequalities are ensured by the $\mathbf{E}'$-step and M-step. $\mathbf{E}'$-step sets $q^{t+1} = \arg\max_{q \in \mathcal{Q}} F(q,\theta^t)$, hence $J_\mathcal{Q}(\theta^t) = F(q^{t+1},\theta^t)$. The M-step sets $\theta^{t+1} = \arg\max_\theta F(q^{t+1},\theta)$, hence $F(q^{t+1},\theta^{t+1}) \geq F(q^{t+1},\theta^t)$. Finally, $J_\mathcal{Q}(\theta^{t+1}) = \max_{q \in \mathcal{Q}} F(q,\theta^{t+1}) \geq F(q^{t+1},\theta^{t+1})$ ■

Note that the proposition is only meaningful when $\mathcal{Q}$ is non-empty and $J_\mathcal{Q}$ is well-defined. As for standard EM, to prove that coordinate ascent on $F(q,\theta)$ converges to stationary points of $J_\mathcal{Q}(\theta)$, we need to make additional assumptions on the regularity of the likelihood function and boundedness of the parameter space as in Tseng (2004). This analysis can be easily extended to our setting, but is beyond the scope of the current paper.

We can use the dual formulation of Proposition 2.1 to perform the projection. Proposition 2.2 implies that we can use the same algorithms to perform inference in our projected model q as we did in our original model p_θ. We illustrate this with the running example.

Running Example *For the POS tagging example with zero slack, the optimization problem we need to solve is:*

$$\arg\min_{q} \quad \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \qquad \text{s.t.} \quad \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] \leq -\mathbf{1}$$

where $\mathbf{1}$ *is a vector of with 1 in each entry. The dual formulation is given by*

$$\arg\max_{\lambda \geq 0} \quad \mathbf{1}\cdot\lambda - \log Z(\lambda) \quad \textit{with} \quad q^*(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{X})\exp\{-\lambda^*\cdot\phi(\mathbf{X},\mathbf{Y})\}}{Z(\lambda^*)}. \tag{9}$$

We can solve the dual optimization problem by projected gradient ascent. The HMM model can be factored as products over sentences, and each sentence as a product of emission probabilities and transition probabilities.

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \frac{\prod_{i=1}^{|\mathbf{x}|} p_\theta(y_i|y_{i-1})p_\theta(x_i|y_i)}{p_\theta(\mathbf{x})} \tag{10}$$

where $p_\theta(y_1|y_0) = p_\theta(y_1)$ *are the initial probabilities of our HMM. The constraint features* ϕ *can be represented as a sum over sentences and further as a sum over positions in the sentence:*

$$\phi(\mathbf{x},\mathbf{y}) \quad = \quad \sum_{i=1}^{|\mathbf{x}|}\phi_i(\mathbf{x},y_i) \quad = \quad \sum_{i=1}^{|\mathbf{x}|}\begin{cases}(-1,0)^\top & \textit{if } y_i \textit{ is a verb in sentence } \mathbf{x}\\ (0,-1)^\top & \textit{if } y_i \textit{ is a noun in sentence } \mathbf{x}\\ (0,0)^\top & \textit{otherwise}\end{cases} \tag{11}$$

combining the factored Equations 10 and 11 with the definition of $q(\mathbf{Y})$ *we see that* $q(\mathbf{Y})$ *must also factor as a first-order Markov model for each sentence:*

$$q^*(\mathbf{Y}) \propto \quad \prod_{\mathbf{x}\in\mathbf{X}}\prod_{i=1}^{|\mathbf{x}|} p_\theta(y_i|y_{i-1})p_\theta(x_i|y_i)e^{-\lambda^*\cdot\phi_i(\mathbf{x},y_i)}.$$

Hence $q^*(\mathbf{Y})$ *is just a first-order Markov model for each sentence, and we can compute the normalizer* $Z(\lambda^*)$ *and marginals* $q(y_i)$ *for each example using forward-backward. This allows computation of the dual objective in Equation 9 as well as its gradient efficiently. The gradient of the dual objective is* $1 - \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]$*. We can use projected gradient (Bertsekas, 1999) to perform the optimization, and the projection can be done sentence-by-sentence allowing for online optimization such as stochastic gradient. Optimization for non-zero slack case can be done using projected subgradient (since the norm is not smooth).*

Note that on *unseen* unlabeled data, the learned parameters θ might not satisfy the constraints on posteriors exactly, although typically they are fairly close if the model has enough capacity.

2.7 Penalized Slack via Expectation Maximization

If our objective is specified using slack-penalty such as in Equation 3, then we need a slightly different E-step. Instead of restricting $q \in Q$, the modified $\mathbf{E}'$-step adds a cost for violating the constraints

$$\begin{aligned}\mathbf{E}' : \min_{q,\xi} \quad & \mathbf{KL}\left(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})\right) + \sigma||\xi||_\beta \\ \text{s.t.} \quad & \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi.\end{aligned} \tag{12}$$

An analogous monotonic improvement of modified EM can be shown for the slack-penalized objective. The dual of Equation 12 is

$$\max_{\lambda \geq 0} \; -\mathbf{b} \cdot \lambda - \log Z(\lambda) \qquad \text{s.t.} \quad ||\lambda||_{\beta^*} \leq \sigma.$$

2.8 PR for Discriminative Models

The PR framework can be used to guide learning in discriminative models as well as generative models. In the case of a discriminative model, we only have $p_\theta(\mathbf{y}|\mathbf{x})$, and the likelihood does not depend on unlabeled data. Specifically,

$$\mathcal{L}^D(\theta) = \log p_\theta(\mathbf{Y}_L|\mathbf{X}_L) + \log p(\theta),$$

where $(\mathbf{Y}_L, \mathbf{X}_L)$ are any available labeled data and $\log p(\theta)$ is a prior on the model parameters. With this definition of $\mathcal{L}(\theta)$ for discriminative models we will optimize the discriminative PR objective (zero-slack case):

$$\textbf{Discriminative PR Likelihood}: \quad J^D_Q(\theta) = \mathcal{L}^D(\theta) - \mathbf{KL}(Q \,\|\, p_\theta(\mathbf{Y}|\mathbf{X})). \tag{13}$$

In the absence of both labeled data and a prior on parameters $p(\theta)$, the objective in Equation 2 is optimized (equal to zero) for any $p_\theta(\mathbf{Y} \mid \mathbf{X}) \in Q$. If we employ a parametric prior on θ, then we will prefer parameters that come close to satisfying the constraints, where proximity is measured by KL-divergence.

> **Running Example** *For the POS tagging example, our discriminative model might be a first order conditional random field. In this case we model:*
>
> $$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{\exp\{\theta \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}}{Z_\theta(\mathbf{x})}$$
>
> *where* $Z_\theta(\mathbf{x}) = \sum_{\mathbf{y}} \exp\{\theta \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}$ *is a normalization constant and* $\mathbf{f}(\mathbf{x}, \mathbf{y})$ *are the* model *features. We will use the same constraint features as in the generative case:* $\phi(\mathbf{x}, \mathbf{y}) =$ *"negative number of verbs in* $\mathbf{y}$*", and define* $Q_{\mathbf{x}}$ *and* $q_{\mathbf{x}}$ *also as before. Note that* $\mathbf{f}$ *are features used to define the model and do not appear anywhere in the constraints while* ϕ *are constraint features that do not appear anywhere in the model.*

Traditionally, the EM algorithm is used for learning generative models (the model can condition on a subset of observed variables, but it must define a distribution over some observed variables). The reason for this is that EM optimizes marginal log-likelihood ($\mathcal{L}$ in our notation) of the observed data $\mathbf{X}$ according to the model. In the case of a discriminative model, $p_\theta(\mathbf{Y}|\mathbf{X})$, we do not model the distribution of the observed data, the value of $\mathcal{L}^D$ as a function of θ depends only on the parametric prior $p(\theta)$ and the labeled data. By contrast, the PR objective uses the KL term and the corresponding constraints to bias the model parameters. These constraints depend on the observed data $\mathbf{X}$ and if they are sufficiently rich and informative, they can be used to train a discriminative model. In the extreme case, consider a constraint set Q that contains only a single distribution q, with $q(\mathbf{Y}^*) = 1$. So, q is concentrated on a particular labeling $\mathbf{Y}^*$. In this case, the PR objective in Equation 13 reduces to

$$J^D_Q(\theta) = \mathcal{L}^D(\theta) + \log p_\theta(\mathbf{Y}^*|\mathbf{X}) = \log p(\theta) + \log p_\theta(\mathbf{Y}_L|\mathbf{X}_L) + \log p_\theta(\mathbf{Y}^*|\mathbf{X}).$$

§#	Problem	Gen/Disc	p/q	Summary of Structural Constraints
§5	Word Alignment	G	q	Translation process is symmetric and bijective
§6	Multi-view learning	D	q	Multiple views should agree on label distribution
§7	Dependency Parsing	G+D	p	Noisy, partially observed labels encoded in ϕ and $\mathbf{b}$
§8	Part-of-speech induction	G	p	Sparsity structure independent of model parameters: each word should be generated by a small number of POS tags

Table 2: Summary of applications of Posterior Regularization described in this paper. Gen/Disc refers to generative or discriminative models. The p/q column shows whether we use the original model p or the projected distribution q at decode time.

Thus, if $\mathcal{Q}$ is informative enough to uniquely specify a labeling of the unlabeled data, the PR objective reduces to the supervised likelihood objective. When $\mathcal{Q}$ specifies a range of distributions, such as the one for multi view learning (Section 6), PR biases the discriminative model to have $p_\theta(\mathbf{Y}|\mathbf{X})$ close to $\mathcal{Q}$.

Equation 13 can also be optimized with a block-coordinate ascent, leading to an EM style algorithm very similar to the one presented in Section 2.6. We define a lower bounding function:

$$F'(q,\theta) = -\mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) = \sum_{\mathbf{Y}} q(\mathbf{Y}) \log \frac{p_\theta(\mathbf{Y}|\mathbf{X})}{q(\mathbf{Y})}.$$

Clearly, $\max_{q\in\mathcal{Q}} F'(q,\theta) = -\mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X}))$ so $F'(q,\theta) \leq -\mathbf{KL}(\mathcal{Q} \parallel p_\theta(\mathbf{Y}|\mathbf{X}))$ for $q \in \mathcal{Q}$.

The modified $\mathbf{E}'$ and $\mathbf{M}'$ steps are:[6]

$$\begin{aligned}
\mathbf{E}' : q^{t+1} &= \arg\max_{q\in\mathcal{Q}} F'(q,\theta^t) = \arg\min_{q\in\mathcal{Q}} \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y}|\mathbf{X})), \\
\mathbf{M}' : \theta^{t+1} &= \arg\max_{\theta} F'(q^{t+1},\theta) = \arg\max_{\theta} \mathbf{E}_{q^{t+1}}\left[\log p_\theta(\mathbf{Y}|\mathbf{X})\right].
\end{aligned} \tag{14}$$

Here the difference between Equation 7 and Equation 14 is that now there is no generative component in the lower-bound $F'(q,\theta)$ and hence we have a discriminative update to the model parameters in Equation 14.

3. Summary of Applications

Because the PR framework allows very diverse prior information to be specified in a single formalism, the application Sections (§5-§8) are very diverse in nature. This section attempts to summarize their similarities and differences without getting into the details of the problem applications and intuition behind the constraints. Table 2 summarizes the applications and constraints described in the rest of the paper while Table 3 summarizes the meanings of the variables $\mathbf{x}$, $\mathbf{y}$ and $\phi(\mathbf{X},\mathbf{Y})$ as well as the optimization procedures used for the applications presented in the sequel.

In the statistical word alignment application described in Section 5, the goal is to identify pairs or sets of words that are direct translations of each other. The statistical models used suffer from what

6. As with the **M-step** in Equation 7 we have ignored the prior $p(\theta)$ on model parameters and the labeled data terms, which can be easily incorporated in the $\mathbf{M}'$ step.

Application	Symbol	Meaning
Word Alignment	$\mathbf{x}$	Pair of sentences that are translations of each other.
	$\mathbf{y}$	Set of alignment links between words in the sentences (exponentially many possibilities).
	$\phi(\mathbf{x},\mathbf{y})$	Bijective: number of times each source word is used in the alignment $\mathbf{y}$. Symmetric: expected difference in number of times each link is used by source$\rightarrow$target model and target $\rightarrow$source model.
	OPT	Bijective: projected gradient. Symmetric: L-BFGS.
Multi-view learning	$\mathbf{x}$	Varies by application.
	$\mathbf{y}$	Varies by application.
	$\phi(\mathbf{x},\mathbf{y})$	Indexed by label; ± 1 if only one model predicts the label, and 0 if both or none predict the label.
	OPT	Closed form.
Dependency Parsing	$\mathbf{x}$	Natural language sentence as a sequence of words.
	$\mathbf{y}$	Dependency parse tree (set of edges from one word to another, forming a tree).
	$\phi(\mathbf{x},\mathbf{y})$	Number of edges in $\mathbf{y}$ that are in the set of translated edges.
	OPT	Line search.
Part-of-speech induction	$\mathbf{x}$	Natural language sentence as a sequence of words.
	$\mathbf{y}$	Sequence of syntactic categories, one for each word.
	$\phi(\mathbf{X},\mathbf{Y})$	Indexed by w,i,s; 1 if the i^{th} occurrence of word w in the corpus is tagged with syntactic category s, and 0 otherwise.
	OPT	Projected gradient.

Table 3: Summary of input and output variable meanings as well as meanings of constraint features and optimization methods used (OPT) for the applications summarized in Table 2.

is known as a garbage collector effect: the likelihood function of the simplistic translation models used prefers to align sections that are not literal translations to rare words, rather than leaving them unaligned (Brown et al., 1993). This results in each rare word in a source language being aligned to 4 or 5 words in the target language. To alleviate this problem, we introduce constraint features that count how many target words are aligned to each source word, and use PR to encourage models where this number is small in expectation. Modifying the model itself to include such a preference would break independence and make it intractable.

The multi-view learning application described in Section 6 leverages two or more sources of input ("views") along with unlabeled data. The requirement is to train two models, one for each view, such that they usually agree on the labeling of the unlabeled data. We can do this using PR, and we recover the Bhattacharyya distance as a regularizer. The PR approach also extends naturally to structured problems, and cases where we only want partial agreement.

The grammar induction application of Section 7 takes advantage of an existing parser for a resource-rich language to train a comparable resource for a resource-poor language. Because the two languages have different syntactic structures, and errors in word alignment abound, using such out-of-language information requires the ability to train with missing and noisy labeling. This is

achieved using PR constraints that guide learning to prefer models that tend to agree with the noisy labeling wherever it is provided, while standard regularization guides learning to be self-consistent.

Finally, Section 8 describes an application of PR to ensure a particular sparsity structure, which can be independent of the structure of the model. Section 8 focuses on the problem of unsupervised part-of-speech induction, where we are given a sample of text and are required to specify a syntactic category for each token in the text. A well-known but difficult to capture piece of prior knowledge for this problem is that each word type should only occur with a small number of syntactic categories, even though there are some syntactic categories that occur with many different word types. By using an ℓ_1/ℓ_∞ norm on constraint features we are able to encourage the model to have precisely this kind of sparsity structure, and greatly increase agreement with human-generated syntactic categories.

Table 2 also shows for each application whether we use the distribution over hidden variables given by the model parameters $p_\theta(\mathbf{Y}|\mathbf{X})$ to decode, or whether we first project the distribution to the constraint set and use $q(\mathbf{Y})$ to decode. In general we found that when applying the constraints on the labeled data is sensible, performing the projection before decoding tends to improve performance. For the word alignment application and the multi-view learning application we found decoding with the projected distribution improved performance. By contrast, for dependency parsing, we do not have the English translations at test time and so we cannot perform a projection. For part-of-speech induction the constraints are over the entire corpus, and different regularization strengths might be needed for the training and test sets. Since we did not want to tune a second hyperparameter, we instead decoded with p.

4. Related Frameworks

The work related to learning with constraints on posterior distributions is described in chronological order in the following three subsections. An overall summary is most easily understood in reverse chronological order though, so we begin with a few sentences detailing the connections to it in that order. Liang et al. (2009) describe how we can view constraints on posterior distributions as measurements in a Bayesian setting, and note that inference using such information is intractable. By approximating this problem, we recover either the generalized expectation constraints framework of Mann and McCallum (2007), or with a further approximation we recover a special case of the posterior regularization framework presented in Section 2. Finally, a different approximation recovers the constraint driven learning framework of Chang et al. (2007). To the best of our knowledge, we are the first to describe all these connections.

4.1 Constraint Driven Learning

Chang et al. (2007, 2008) describe a framework called constraint driven learning (CODL) that can be viewed as an approximation to optimizing the slack-penalized version of the PR objective (Equation 3). Chang et al. (2007) are motivated by hard-EM, where the distribution q is approximated by a single sample at the mode of $\log p_\theta(\mathbf{Y}|\mathbf{X})$. Chang et al. (2007) propose to augment $\log p_\theta(\mathbf{Y}|\mathbf{X})$ by adding to it a penalty term based on some domain knowledge. When the penalty terms are well-behaved, we can view them as adding a cost for violating expectations of constraint features ϕ. In such a case, CODL can be viewed as a "hard" approximation to the PR objective:

$$\arg\max_\theta \ \mathcal{L}(\theta) \quad -\min_{q \in M} \Big(\mathbf{KL}(q(\mathbf{Y})||p_\theta(\mathbf{Y}|\mathbf{X})) \ + \ \sigma \big|\big|\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b}\big|\big|_\beta \Big)$$

where M is the set of distributions concentrated on a single $\mathbf{Y}$. The modified E-Step becomes:

$$\text{CODL } \mathbf{E}'\text{-step} : \max_{\mathbf{Y}} \log p_\theta(\mathbf{Y}|\mathbf{X}) \; - \; \sigma||\phi(\mathbf{X},\mathbf{Y}) - \mathbf{b}||_\beta \, .$$

Because the constraints used by Chang et al. (2007) do not allow tractable inference, they use a beam search procedure to optimize the min-KL problem. Additionally they consider a K-best variant where instead of restricting themselves to a single point estimate for q, they use a uniform distribution over the top K samples.

Carlson et al. (2010) train several named entity and relation extractors concurrently in order to satisfy type constraints and mutual exclusion constraints. Their algorithm is related to CODL in that hard assignments are made in a way that guarantees the constraints are satisfied. However, their algorithm is motivated by adding these constraints to algorithms that learn pattern extractors: at each iteration, they make assignments only to the highest confidence entities, which are then used to extract high confidence patterns for use in subsequent iterations. By contrast hard EM and CODL would make assignments to every instance and change these assignments over time. Daumé III (2008) also use constraints to filter out examples for self-training and also do not change the labels.

4.2 Generalized Expectation Criteria

Generalized expectation criteria (GE) allow a user to specify preferences about model expectations in the form of linear constraints on some feature expectations (Mann and McCallum, 2007, 2008). As with PR, a set of constraint features ϕ are introduced, and a penalty term is added to the log-likelihood objective. The GE objective is

$$\max_{\theta} \; \mathcal{L}(\theta) - \sigma||\mathbf{E}_{p_\theta}[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b}||_\beta \, . \tag{15}$$

where $||\cdot||_\beta$ is typically the l_2 norm (Druck et al., 2009 use l_2^2) or a distance based on KL divergence (Mann and McCallum, 2008), and the model is a log-linear model such as maximum entropy or a CRF.

The idea leading to this objective is the following: Suppose that we only had enough resources to make a very small number of measurements when collecting statistics about the true distribution $p^*(\mathbf{y}|\mathbf{x})$. If we try to create a maximum entropy model using these statistics we will end up with a very impoverished model. It will only use a small number of features and consequently will fail to generalize to instances where these features cannot occur. In order to train a more feature-rich model, GE defines a wider set of *model* features $\mathbf{f}$ and uses the small number of estimates based on constraint features ϕ to guide learning. By using l_2 regularization on model parameters, we can ensure that a richer set of model features are used to explain the desired expectations.

Druck et al. (2009) use a gradient ascent method to optimize the objective. Unfortunately, because the second term in the GE objective (Equation 15) couples the constraint features ϕ and the model parameters θ, the gradient requires computing the covariance between model features $\mathbf{f}$ and the constraint features ϕ under p_θ:

$$\frac{\partial \mathbf{E}_{p_\theta}[\phi(\mathbf{X},\mathbf{Y})]}{\partial \theta} = \mathbf{E}_{p_\theta}[\mathbf{f}(\mathbf{X},\mathbf{Y})\phi(\mathbf{X},\mathbf{Y})] - \mathbf{E}_{p_\theta}[\phi(\mathbf{X},\mathbf{Y})]\mathbf{E}_{p_\theta}[\mathbf{f}(\mathbf{X},\mathbf{Y})].$$

Because of this coupling, the complexity of the dynamic program needed to compute the gradient is higher than the complexity of the dynamic program for the model. In the case of graphical models

where $\mathbf{f}$ and ϕ have the same Markov dependencies, computing this gradient usually squares the running time of the dynamic program. A more efficient dynamic program might be possible (Li and Eisner, 2009; Pauls et al., 2009), however current implementations are prohibitively slower than PR when there are many constraint features.

In order to avoid the costly optimization procedure described above, Bellare et al. (2009) propose a variational approximation. Recall that at a high level, the difficulty in optimizing Equation 15 is because the last term couples the constraint features ϕ with the model parameters θ. In order to separate out these quantities, Bellare et al. (2009) introduce an auxiliary distribution $q(\mathbf{Y}) \approx p_\theta(\mathbf{Y}|\mathbf{X})$, which is used to approximate the last term in Equation 15. The variational objective contains three terms instead of two:

$$\arg\max_{\theta} \ \mathcal{L}(\theta) \quad - \min_{q} \Big(\mathbf{KL}(q(\mathbf{Y})||p_\theta(\mathbf{Y}|\mathbf{X})) \ + \ \sigma \big|\big|\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b}\big|\big|_\beta \Big). \tag{16}$$

This formulation is identical to the slack-penalized version of PR, and Bellare et al. (2009) use the same optimization procedure (described in Section 2). Because both the minorization and the maximization steps implement minimum Kullback-Leibler projections, Bellare et al. (2009) refer to this algorithm as alternating projections. Note that PR can also be trained in an online fashion, and Ganchev et al. (2009) use an online optimization for this objective to train a dependency parser. These experiments are described in Section 7.

Closely related to GE, is the work of Quadrianto et al. (2009). The authors describe a setting where the constraint values, $\mathbf{b}$, are chosen as the empirical estimates on some labeled data. They then train a model to have high likelihood on the labeled data, but also match the constraint features on unlabeled data. They show that for appropriately chosen constraint features, the estimated constraint values should be close to the true means, and show good experimental improvements on an image retrieval task.

4.3 Measurements in a Bayesian Framework

Liang et al. (2009) approach the problem of incorporating prior information about model posteriors from a Bayesian point of view. They motivate their approach using the following caricature. Suppose we have log-linear model $p_\theta(\mathbf{y}|\mathbf{x}) \propto \exp(\theta \cdot \mathbf{f}(\mathbf{y},\mathbf{x}))$. In addition to any labeled data $(\mathbf{X}_L, \mathbf{Y}_L)$, we also have performed some additional experiments.[7] In particular, we have observed the expected values of some constraint features $\phi(\mathbf{X},\mathbf{Y})$ on some unlabeled data $\mathbf{X}$. Because there is error in measurement, they observe $\mathbf{b} \approx \phi(\mathbf{X},\mathbf{Y})$. Figure 3 illustrates this setting. The leftmost nodes represent $(\mathbf{x},\mathbf{y})$ pairs from the labeled data $(\mathbf{X}_L, \mathbf{Y}_L)$. The nodes directly to the right of θ represent unlabeled $(\mathbf{x},\mathbf{y})$ pairs from the unlabeled data $\mathbf{X}$. All the data are tied together by the dependence on the model parameters θ. The constraint features take as input the unlabeled data set $\mathbf{X}$ as well as a full labeling $\mathbf{Y}$, and produce some value $\phi(\mathbf{X},\mathbf{Y})$, which is never observed directly. Instead, we observe some noisy version $\mathbf{b} \approx \phi(\mathbf{X},\mathbf{Y})$. The measured values $\mathbf{b}$ are distributed according to some noise model $p_N(\mathbf{b}|\phi(\mathbf{X},\mathbf{Y}))$. Liang et al. (2009) note that the optimization is convex for log-concave noise and use box noise in their experiments, giving $\mathbf{b}$ uniform probability in some range near $\phi(\mathbf{X},\mathbf{Y})$.

In the Bayesian setting, the model parameters θ as well as the observed measurement values $\mathbf{b}$ are random variables. Liang et al. (2009) use the mode of $p(\theta|\mathbf{X}_L,\mathbf{Y}_L,\mathbf{X},\mathbf{b})$ as a point estimate

7. In their exposition, Liang et al. (2009) incorporate labeled data by including the labels among experiments. We prefer to separate these types of observations because full label observations do not require approximations.

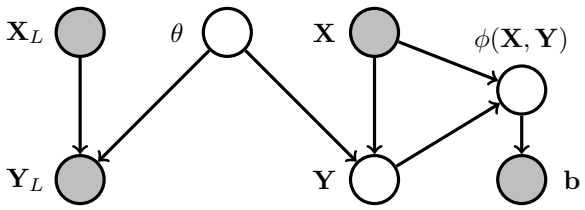


Figure 3: The model used by Liang et al. (2009), using our notation. We have separated treatment of the labeled data $(\mathbf{X}_L, \mathbf{Y}_L)$ from treatment of the unlabeled data $\mathbf{X}$.

for θ:

$$\arg\max_{\theta} \; p(\theta|\mathbf{X}_L, \mathbf{Y}_L, \mathbf{X}, \mathbf{b}) \quad = \quad \arg\max_{\theta} \; \sum_{\mathbf{Y}} p(\theta, \mathbf{Y}, \mathbf{b}|\mathbf{X}, \mathbf{X}_L, \mathbf{Y}_L),$$

with equality because $p(\theta|\mathbf{X}_L, \mathbf{Y}_L, \mathbf{X}, \mathbf{b}) \propto p(\theta, \mathbf{b}|\mathbf{X}_L, \mathbf{Y}_L, \mathbf{X}) = \sum_{\mathbf{Y}} p(\theta, \mathbf{Y}, \mathbf{b}|\mathbf{X}, \mathbf{X}_L, \mathbf{Y}_L)$. Liang et al. (2009) focus on computing $p(\theta, \mathbf{Y}, \mathbf{b}|\mathbf{X}, \mathbf{X}_L, \mathbf{Y}_L)$. They define their model for this quantity as follows:

$$p(\theta, \mathbf{Y}, \mathbf{b}|\mathbf{X}, \mathbf{X}_L, \mathbf{Y}_L) \quad = \quad p(\theta|\mathbf{X}_L, \mathbf{Y}_L) \;\; p_\theta(\mathbf{Y}|\mathbf{X}) \;\; p_N(\mathbf{b}|\phi(\mathbf{X}, \mathbf{Y})) \tag{17}$$

where the $\mathbf{Y}$ and $\mathbf{X}$ are particular instantiations of the random variables in the entire unlabeled corpus $\mathbf{X}$. Equation 17 is a product of three terms: a prior on θ, the model probability $p_\theta(\mathbf{Y}|\mathbf{X})$, and a noise model $p_N(\mathbf{b}|\phi)$. The noise model is the probability that we observe a value, $\mathbf{b}$, of the measurement features ϕ, given that its actual value was $\phi(\mathbf{X}, \mathbf{Y})$. The idea is that we model errors in the estimation of the posterior probabilities as noise in the measurement process. Liang et al. (2009) use a uniform distribution over $\phi(\mathbf{X}, \mathbf{Y}) \pm \varepsilon$, which they call "box noise". Under this model, observing $\mathbf{b}$ farther than ε from $\phi(\mathbf{X}, \mathbf{Y})$ has zero probability. In log space, the exact MAP objective, becomes:

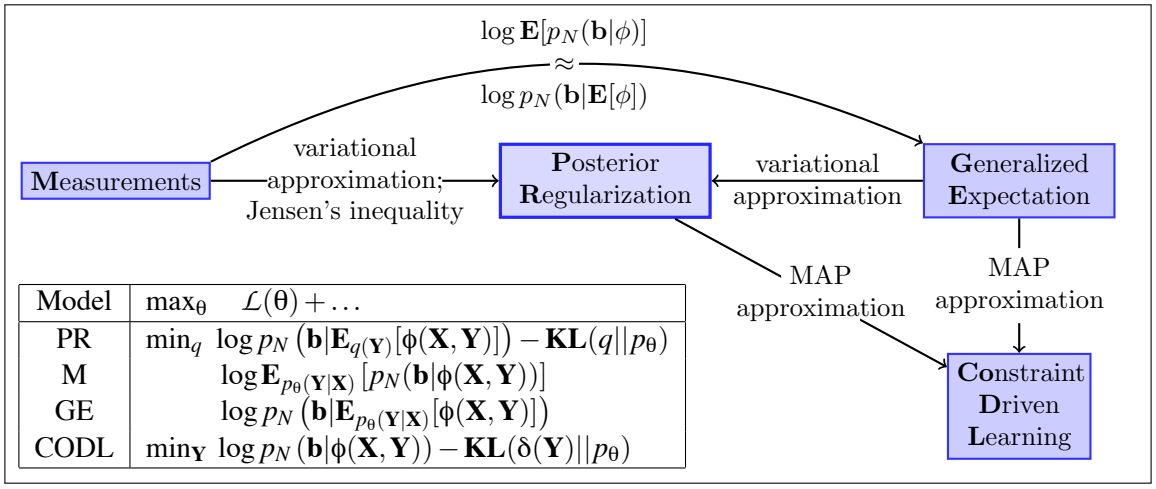
$$\max_{\theta} \quad \mathcal{L}(\theta) \quad + \log \mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})} \left[p_N(\mathbf{b}|\phi(\mathbf{X}, \mathbf{Y})) \right]. \tag{18}$$

Unfortunately with almost all noise models (including no noise), and box noise in particular, the second term in Equation 18 makes the optimization problem intractable.[8] Liang et al. (2009) use a variational approximation as well as a further approximation based on Jensen's inequality to reach the PR objective, which they ultimately optimize for their experiments. We also relate their framework to GE and CODL. If we approximate the last term in Equation 18 by moving the expectation inside the probability:

$$\mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})} \left[p_N(\mathbf{b} \mid \phi(\mathbf{X}, \mathbf{Y})) \right] \approx p_N \left(\mathbf{b} \mid \mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})} [\phi(\mathbf{X}, \mathbf{Y})] \right),$$

we end up with an objective equivalent to GE for appropriate noise models. In particular Gaussian noise corresponds to l_2^2 regularization in GE, since the log of a Gaussian is squared Euclidean distance (up to scaling). This approximation can be motivated by the case when $p_\theta(\mathbf{Y}|\mathbf{X})$ is concentrated on a particular labeling $\mathbf{Y}^*$: $p_\theta(\mathbf{Y}|\mathbf{X}) = \delta(\mathbf{Y}^*)$. In this special case the $\approx$ is an equality.

8. For very special noise, such as noise that completely obscures the signal, we can compute the second term in Equation 18.



Model	$\max_\theta \quad \mathcal{L}(\theta) + \ldots$
PR	$\min_q \ \log p_N\left(\mathbf{b}\middle\vert\mathbf{E}_{q(\mathbf{Y})}[\phi(\mathbf{X},\mathbf{Y})]\right) - \mathbf{KL}(q\vert\vert p_\theta)$
M	$\log \mathbf{E}_{p_\theta(\mathbf{Y}\vert\mathbf{X})}\left[p_N(\mathbf{b}\vert\phi(\mathbf{X},\mathbf{Y}))\right]$
GE	$\log p_N\left(\mathbf{b}\middle\vert\mathbf{E}_{p_\theta(\mathbf{Y}\vert\mathbf{X})}[\phi(\mathbf{X},\mathbf{Y})]\right)$
CODL	$\min_\mathbf{Y} \ \log p_N\left(\mathbf{b}\middle\vert\phi(\mathbf{X},\mathbf{Y})\right) - \mathbf{KL}(\delta(\mathbf{Y})\vert\vert p_\theta)$

Figure 4: A summary of the different models. We use $p_\theta(\mathbf{Y}|\mathbf{X})$ to denote the model probability, $q(\mathbf{Y})$ to denote a proposal distribution, and p_N for the noise model. The symbol $\delta(\mathbf{Y})$ denotes a distribution concentrated on $\mathbf{Y}$. The approximations are described in the text: M→GE near Equation 19, GE→PR near Equation 16, PR→CODL at the end of Section 4.3.

This approximation is also used in Liang et al. (2009). This provides an interpretation of GE as an approximation to the Bayesian framework proposed by Liang et al. (2009):

$$\max_\theta \quad \mathcal{L}(\theta) \quad + \log p_N\left(\mathbf{b}\middle|\mathbf{E}_{p_\theta(\mathbf{Y}|\mathbf{X})}[\phi(\mathbf{X},\mathbf{Y})]\right). \tag{19}$$

Note that the objective in Equation 19 is a reasonable objective in and of itself, essentially stating that the measured values $\mathbf{b}$ are not dependent on any particular instantiation of the hidden variables, but rather represent the integral over all their possible assignments. Liang et al. (2009) also use a variational approximation similar to the one of Bellare et al. (2009) so that the objective they optimize is exactly the PR objective, although their optimization algorithm is slightly different from the one presented in Section 2. Finally, if we restrict the set of allowable distributions further to be concentrated on a single labeling $\mathbf{Y}$, we recover the CODL algorithm. Figure 4 summarizes the relationships.

5. Statistical Word Alignments

Word alignments, introduced by Brown et al. (1994) as hidden variables in probabilistic models for statistical machine translation (IBM models 1-5), describe the correspondence between words in source and target sentences. We will denote each target sentence as $\mathbf{x}^t = (x_1^t, \ldots, x_i^t, \ldots, x_I^t)$ and each source sentence as $\mathbf{x}^s = (x_1^s, \ldots, x_j^s, \ldots, x_J^s)$. A word alignment will be represented as a matrix with entries y_{ij} indicating that target word i is a translation of source word j. Although the original IBM models are no longer competitive for machine translation, the resulting word alignments are still a valuable resource. Word alignments are used primarily for extracting minimal translation units for machine translation, for example, phrases in phrase-based translation systems (Koehn et al., 2003)

and rules in syntax-based machine translation (Galley et al., 2004; Chiang et al., 2005), as well as for MT system combination (Matusov et al., 2006). But their importance has grown far beyond machine translation: for instance, transferring annotations between languages by projecting POS taggers, NP chunkers and parsers through word alignment (Yarowsky and Ngai, 2001; Rogati et al., 2003; Hwa et al., 2005; Ganchev et al., 2009); discovery of paraphrases (Bannard and Callison-Burch, 2005; Callison-Burch, 2007, 2008); and joint unsupervised POS and parser induction across languages (Snyder and Barzilay, 2008; Snyder et al., 2009).

Here we describe two types of prior knowledge that when introduced as constraints in different word alignment models significantly boost their performance. The two constraints are: (i) bijectivity: "one word should not translate to many words"; and (ii) symmetry: "directional alignments of one model should agree with those of another model". A more extensive description of these constraints applied to the task of word alignments and the quality of the resulting alignments can be found in Graça et al. (2010).

5.1 Models

We consider two models below: IBM Model 1 proposed by Brown et al. (1994) and the HMM model proposed by Vogel et al. (1996). Both models can be expressed as:

$$p(\mathbf{x}^t, \mathbf{y} \mid \mathbf{x}^s) = \prod_j p_d(y_j \mid j, y_{j-1}) p_t(\mathbf{x}^t_j \mid \mathbf{x}^s_{y_j}),$$

where $\mathbf{y}$ is the alignment and y_j is the index of the hidden state (source language index) generating the target language word at index j. The models differ in their definition of the distortion probability $p_d(y_j \mid j, y_{j-1})$. Model 1 assumes that the target words are generated independently and assigns uniform distortion probability. The HMM model assumes that only the distance between the current and previous source word index is important $p_d(y_j \mid j, y_{j-1}) = p_d(y_j \mid y_j - y_{j-1})$. Both models are augmented by adding a special "null" word to the source sentence.

The likelihood of the corpus, marginalized over possible alignments, is concave for Model 1, but not for the HMM model (Brown et al., 1994; Vogel et al., 1996). For both models though, standard training using the Expectation Maximization algorithm (Dempster et al., 1977) seeks model parameters θ that maximize the log-likelihood of the parallel corpus.

On the positive side, both models are simple and complexity of inference is $O(I \times J)$ for IBM Model 1 and $O(I \times J^2)$ for the HMM. However there are several problems with the models that arise from their directionality.

- **Non-bijective:** Multiple target words can align to a single source word with no penalty.
- **Asymmetric:** Swapping the source and target languages can produce very different alignments, since only constraints and correlations between consecutive positions on one side are enforced by the models.

The top row of Figure 5 shows an example of the posterior distribution for the alignment between an English and a French sentence using the HMM model. The left figure shows the alignment in the English to French direction where the rows are source words and columns are target words, while the right figure shows the alignment posteriors of the opposite direction. The first observation we make is that the posteriors are concentrated around particular source words (rare words

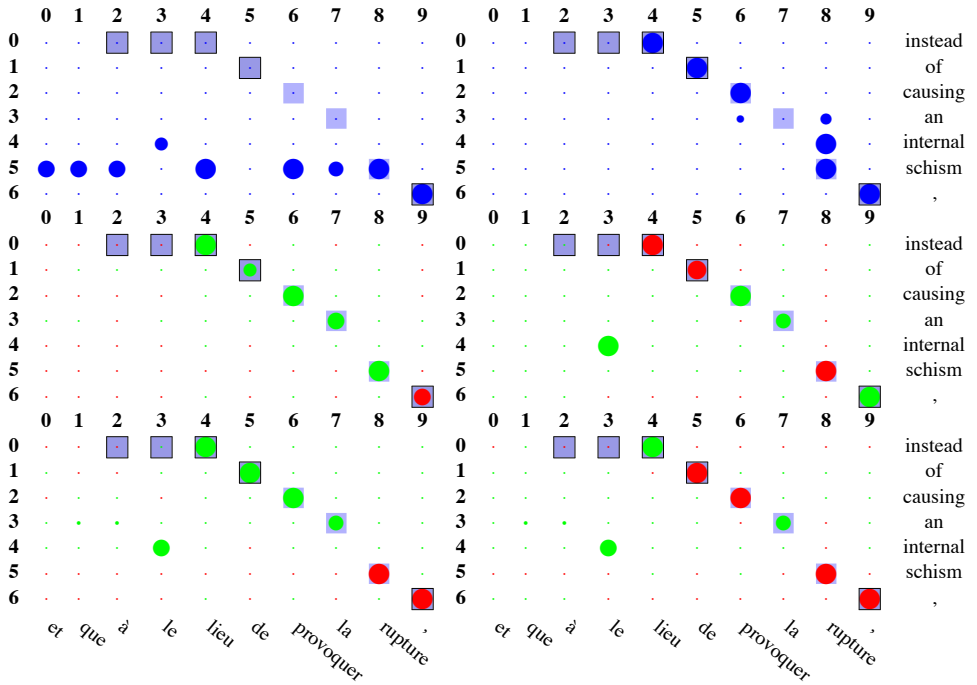


Figure 5: Posterior distributions on an English to French sentence using the HMM model. **Left:** EN→FR model. **Right:** FR→ EN model. **Top:** Regular EM posteriors. **Middle:** After applying bijective constraint. **Bottom:** After applying symmetric constraint. *Sure* alignments are squares with borders; *possible* alignments are squares without borders. Circle size indicates probability value. See Graça et al. (2010) for a description of the difference between sure and possible alignments. Circle color in the middle and bottom rows indicates difference in posterior from the top row. Green (light gray) - higher probability, red (dark gray) - lower probability.

occurring less than 5 times in the corpus) in both directions, instead of being spread across different words. This is a well known problem when training using EM, called the "garbage collector effect" (Brown et al., 1993). That is, rare words in the source language end up aligned to too many words in the target language because the generative model has to distribute translation probability for each source word among all candidate target words. Since the rare source word occurs in only a few sentences it needs to spread its probability mass over fewer competing target words. In this case, choosing to align the rare word to all of these target words leads to higher likelihood than correctly aligning them or aligning them to the special *null* word, since it increases the likelihood of this sentence without lowering the likelihood of many other sentences. Moreover, by correcting the garbage collector effect we increase the overall performance on the common words, since now these common words can be aligned to the correct words. For this particular corpus, 6.5% of the English tokens and 7.7% of the French tokens are rare.

5.2 Bijectivity Constraints

Bijectivity constraints are based on the observation that in most gold alignments, words are aligned one-to-one (98% for the sure alignments in the Hansard corpus). We would like to introduce this

trend into the model, but adding it directly requires large factors (breaking the Markov property). In fact, summing over one-to-one or near one-to-one weighted matchings is a classical #P-Complete problem (Valiant, 1979). However, introducing alignment degree constraints *in expectation* in the PR framework is easy and tractable. We simply add inequality constraints $\mathbf{E}[\phi(\mathbf{x},\mathbf{y})] \leq 1$ where we have one feature for each source word j that counts how many times it is aligned to a target word in the alignment $\mathbf{y}$:

$$\textbf{Bijective Features}: \quad \phi_j(\mathbf{x},\mathbf{y}) = \sum_i \mathbf{1}(y_i = j).$$

For example, in the alignment at the top right of Figure 5, the posteriors over the source word *schism* clearly sum to more than 1. The effect of applying PR constraints to the posteriors is shown in the second row. Enforcing the one to (at most) one constraint clearly alleviates the garbage collector effect. Moreover, when distributing the probability mass to the other words, most of the probability mass goes into the correct positions (as can be seen by comparison to the gold alignments). Note that the bijectivity constraints only hold approximately in practice and so we expect that having a KL-based penalty for them might be better than ensuring that the model satisfies them, since we can violate bijectivity in order to achieve higher likelihood. Another way to understand what is going on is to see how the parameters are affected by the bijectivity constraint. In particular the translation table becomes much cleaner, having a much lower entropy. The average entropy for the translation probability, averaged over all source language words for EM training is 2.0-2.6 bits, depending on the direction. When we train using PR with the bijectivity constraint, this entropy drops to 0.6 for both directions. We see that while the model does not have any particular parameter that can enforce bijectivity, in practice the model is expressive enough to learn a preference for bijectivity by appropriately setting the existing model parameters.

5.3 Symmetry Constraints

Word alignment should not depend on translation direction, but this principle is clearly violated by the directional models. In fact, each directional model makes different mistakes. The standard approach is to train two models independently and then intersect their predictions (Och and Ney, 2003).[9] However, we show that it is much better to train two directional models concurrently, coupling their posterior distributions over alignments with constraints that force them to approximately agree. The idea of training jointly has also been explored by Matusov et al. (2004) and Liang et al. (2006), although their formalization is quite different.

Let the directional models be defined as: $\overrightarrow{p}_\theta(\overrightarrow{\mathbf{y}})$ (forward) and $\overleftarrow{p}_\theta(\overleftarrow{\mathbf{y}})$ (backward). We suppress dependence on $\mathbf{x}^s$ and $\mathbf{x}^t$ for brevity. Define $\mathbf{y}$ to range over the union of all possible directional alignments $\overrightarrow{\mathbf{y}} \cup \overleftarrow{\mathbf{y}}$. We then define a mixture model $p_\theta(\mathbf{y}) = \frac{1}{2}\overrightarrow{p}_\theta(\mathbf{y}) + \frac{1}{2}\overleftarrow{p}_\theta(\mathbf{y})$ where $\overleftarrow{p}_\theta(\overrightarrow{\mathbf{y}}) = 0$ and vice-versa (i.e., the alignment of one directional model has probability zero according to the other model). We then define the following feature for each target-source position pair i, j:

$$\textbf{Symmetric Features}: \quad \phi_{ij}(\mathbf{x},\mathbf{y}) = \begin{cases} +1 & \mathbf{y} \in \overrightarrow{\mathbf{y}} \text{ and } \overrightarrow{y_i} = j \\ -1 & \mathbf{y} \in \overleftarrow{\mathbf{y}} \text{ and } \overleftarrow{y_j} = i \\ 0 & \text{otherwise.} \end{cases}$$

The feature takes the value zero in expectation if a word pair i, j is aligned with equal probability in both directions. So the constraint we want to impose is $\mathbf{E}_q[\phi_{ij}(\mathbf{x},\mathbf{y})] = 0$ (possibly with some small

9. Sometimes union or a heuristic is used instead of intersection.

violation). Note that this constraint is only feasible if the posteriors are bijective. Clearly these features are fully factored, so to compute expectations of these features under the model q we only need to be able to compute them under each directional model, as we show below. To see this, we have by the definition of q_λ and p_θ,

$$q_\lambda(\mathbf{y} \mid \mathbf{x}) = \frac{\overrightarrow{p}_\theta(\mathbf{y} \mid \mathbf{x}) + \overleftarrow{p}_\theta(\mathbf{y} \mid \mathbf{x})}{2} \frac{\exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\}}{Z} = \frac{\overrightarrow{q}(\mathbf{y} \mid \mathbf{x}) \frac{Z_{\overrightarrow{q}}}{\overrightarrow{p}_\theta(\mathbf{x})} + \overleftarrow{q}(\mathbf{y} \mid \mathbf{x}) \frac{Z_{\overleftarrow{q}}}{\overleftarrow{p}_\theta(\mathbf{x})}}{2Z},$$

where we have defined:

$$\begin{aligned}
\overrightarrow{q}(\mathbf{y} \mid \mathbf{x}) = &\quad \frac{1}{Z_{\overrightarrow{q}}} \overrightarrow{p}_\theta(\mathbf{y}, \mathbf{x}) \exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\} \quad \text{with} \quad Z_{\overrightarrow{q}} = \sum_{\mathbf{y}} \overrightarrow{p}_\theta(\mathbf{y}, \mathbf{x}) \exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\}, \\
\overleftarrow{q}(\mathbf{y} \mid \mathbf{x}) = &\quad \frac{1}{Z_{\overleftarrow{q}}} \overleftarrow{p}_\theta(\mathbf{y}, \mathbf{x}) \exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\} \quad \text{with} \quad Z_{\overleftarrow{q}} = \sum_{\mathbf{y}} \overleftarrow{p}_\theta(\mathbf{y}, \mathbf{x}) \exp\{-\lambda \cdot \phi(\mathbf{x}, \mathbf{y})\}, \\
Z = &\quad \frac{1}{2} \left(\frac{Z_{\overrightarrow{q}}}{\overrightarrow{p}_\theta(\mathbf{x})} + \frac{Z_{\overleftarrow{q}}}{\overleftarrow{p}_\theta(\mathbf{x})} \right).
\end{aligned}$$

All these quantities can be computed separately in each model.

The last row in Figure 5 shows both directional posteriors after imposing the symmetric constraint. Note that the projected posteriors are equal in the two models. Also, one can see that in most cases the probability mass was moved to the correct place with the exception of the word pair *internal*/*le*; this is because the word *internal* does not appear on the French side, but the model still has to spread around the probability mass for that word. In this case the model decided to accumulate it on the word *le* instead of moving it to the *null* word.

5.4 Algorithm for Projection

Because both the symmetric and bijective constraints decompose over sentences, and the model distribution $p(\mathbf{Y}|\mathbf{X})$ decomposes as a product distribution over instances, the projected distribution $q(\mathbf{Y})$ will also decompose as a product over instances. Furthermore, because the constraints for different instances do not interact, we can project the sentences one at a time and we do not need to store the posteriors for the whole corpus in memory at once. We compute $q(\mathbf{y})$ for each sentence pair $\mathbf{x}$ using projected gradient ascent for the bijective constraints and L-BFGS for the symmetric constraints.

5.5 Results

We evaluated the constraints using the Hansard corpus (Och and Ney, 2000) of English/French. Following prior work by Och and Ney (2003), we initialize the Model 1 translation table with uniform probabilities over word pairs that occur together in same sentence. The HMM is initialized with the translation probabilities from Model 1 and with uniform distortion probabilities. We train M1 for 5 iterations and train the HMM model until no further improvement on precision and recall is seen on standard (small) development set for this corpus. We note that when using regular EM training this requires around 4 iterations, while just 2 iterations suffices when using PR. This is likely due to the added information that the constraints provide. We use a 40 word maximum length cutoff for training sentences and train all models on 100,000 sentences, testing precision and recall on the standard test set.

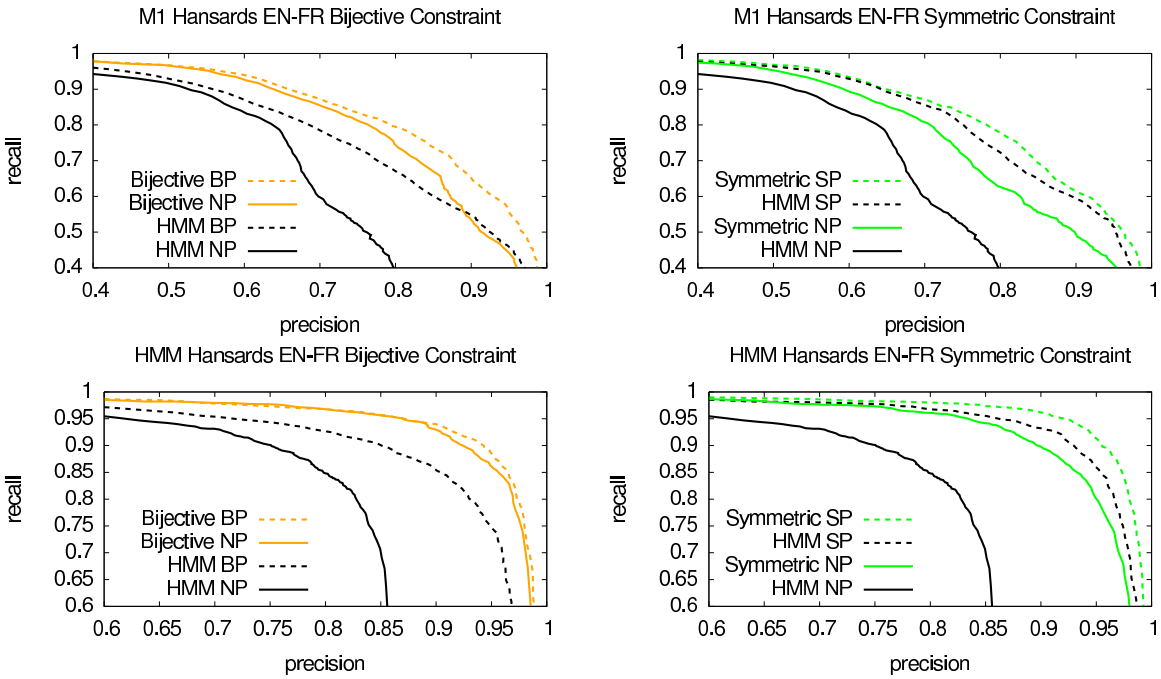


Figure 6: Precision vs Recall curves of both models using standard EM training (Regular) versus PR with bijective constraints (Bijective) and symmetry constraints (Symmetric) and different decoding types: decoding without any projection (NP), doing bijective projection before decoding (BP), and doing symmetric projection before decoding (SP). Data is 100k sentences of the Hansard corpus. Highest label in the legend corresponds to highest line in the graph, second highest label to second highest line, and so on.

Figure 6 shows the precision vs recall curves of both models (EN-FR, FR-EN independently) when training using standard EM versus PR with both constraints, and the results of additionally applying the constraints at decode time in order to tease apart the effect of the constraints during training vs. during testing. The first observation is that training with PR significantly boosts the performance of each model. Moreover using the projection at decode time always increases performance. Comparing both constraints, it seems that the bijective constraint is more useful at training time. Note that using this constraint at decode time with regular training yields worse results than just training with the same constraint using PR. On the other hand, the symmetric constraint is stronger at decode time.

A comprehensive comparison of the word alignment application is presented in Graça et al. (2010), where in six different languages the use of these constraints always significantly outperforms the simpler unconstrained HMM model. Moreover, in 9 out of 12 times using these constraints outperforms the more complex model IBM M4 (Brown et al., 1994). The alignments are also evaluated in the task of statistical machine translation where they are used as an initial step to extract parallel phrases used for translation. Using these constraints leads to better translation quality. Finally, the different alignments are tested on the task of transferring syntactic annotations, which

is described in the next section. The new alignments increase the number of correctly transferred edges.

6. Multi-view learning

Multi-view learning refers to a set of semi-supervised methods which exploit redundant views of the same input data (Blum and Mitchell, 1998; Collins and Singer, 1999; Brefeld et al., 2005; Sindhwani et al., 2005). These multiple views can come in the form of context and spelling features in the case of text processing and segmentation, hypertext link text and document contents for document classification, and multiple cameras or microphones in the case of speech and vision. Multi-view methods typically begin by assuming that each view alone can yield a good predictor. Under this assumption, we can regularize the models from each view by constraining the amount by which we permit them to disagree on unlabeled instances. This regularization can lead to better convergence by significantly decreasing the effective size of our hypothesis class (Balcan and Blum, 2005; Kakade and Foster, 2007; Rosenberg and Bartlett, 2007). This idea is related to the symmetry constraints described in Section 5.

In this section, we use PR to derive a multi-view learning algorithm. The idea is very simple: train a model for each view, and use constraints that the models should agree on the label distribution. Where our work is most similar to co-regularization schemes, a minimum Kullbeck-Leibler (KL) distance projection can be computed in closed form resulting in an algorithm that performs better than both CoBoosting and two view Perceptron on several natural language processing tasks. In this case, the resulting regularizer is identical to adding a penalty term based on the Bhattacharyya distance (Kailath, 1967) between models trained using different views.

In addition, this framework allows us to use different labeled training sets for the two classifiers, in the case where they have different label sets. That is, we don't require that our two views are both on the same labeled corpus. In that case, we can reduce the hypothesis space by preferring pairs of models that agree on *compatible* labeling of some additional unlabeled data rather than on *identical* labeling, while still minimizing KL in closed form. When the two views come from models that differ not only in the label set but also in the model structure of the output space, our framework can still encourage agreement, but the KL minimization cannot be computed in closed form. Finally, this method uses soft assignments to latent variables resulting in a more stable optimization procedure.

6.1 Stochastic Agreement

Note that the constraint in this section is similar to the one described in Section 5, but here we focus on discriminative learning and the formulation is slightly different. For notational convenience, we focus on two view learning in this exposition, however the generalization to more than two views is fairly straightforward. Also, we focus on two discriminative log-linear models and start by considering the setting of complete agreement. In this setting we have a common desired output for the two models and we believe that each of the two views is sufficiently rich to predict labels accurately. We can leverage this knowledge by restricting our search to model pairs p_1,p_2 that satisfy $p_1(\mathbf{y} \mid \mathbf{x}) \approx p_2(\mathbf{y} \mid \mathbf{x})$. Since p_1 and p_2 each define a distribution over labels, we will consider the product distribution $p_1(\mathbf{y}_1)p_2(\mathbf{y}_2)$ and define constraint features such that our proposal distribution $q(\mathbf{y}_1, \mathbf{y}_2)$ will have the same marginal for $\mathbf{y}_1$ and $\mathbf{y}_2$. In particular, we will have one constraint feature for each label $\mathbf{y}$:

$$\phi_{\mathbf{y}}(\mathbf{y}_1, \mathbf{y}_2) = \delta(\mathbf{y}_1 = \mathbf{y}) - \delta(\mathbf{y}_2 = \mathbf{y}).$$

Where $\delta(\cdot)$ is the 0-1 indicator function. The constraint set $Q = \{q : \mathbf{E}_q[\phi] = 0\}$ will require that the marginals over the two output variables are identical $q(\mathbf{y}_1) = q(\mathbf{y}_2)$. It will be useful in the sequel to define an agreement between two models agree(p_1, p_2) as

$$\text{agree}(p_1, p_2) = \arg\min_q \ \mathbf{KL}(q(\mathbf{y}_1, \mathbf{y}_2)||p_1(\mathbf{y}_1)p_2(\mathbf{y}_2)) \quad \text{s.t.} \quad \mathbf{E}_q[\phi] = 0. \tag{20}$$

Proposition 6.1 relates the Bhattacharyya regularization term to the value of the optimization problem in Equation 20. The Bhattacharyya distance is a very natural, symmetric measure of difference between distributions which has been used in many signal detection applications (Kailath, 1967). It is also related to the well-known Hellinger distance.

Proposition 6.1 *The Bhattacharyya distance* $-\log \sum_{\mathbf{y}} \sqrt{p_1(\mathbf{y})p_2(\mathbf{y})}$ *is equal to* $\frac{1}{2}$ *of the value of the convex optimization problem*

$$\begin{array}{ll} \min\limits_{q \in Q} & \mathbf{KL}(q(\mathbf{y}_1, \mathbf{y}_2)||p_1(\mathbf{y}_1)p_2(\mathbf{y}_2)) \\ \text{where} & Q = \left\{q : \mathbf{E}_q[\delta(\mathbf{y}_1 = \mathbf{y}) - \delta(\mathbf{y}_2 = \mathbf{y})] = 0 \ \forall y\right\}, \end{array} \tag{21}$$

and where $\delta(cond)$ *is* 1 *if cond is true and* 0 *otherwise. Furthermore, the minimizer decomposes as* $q(\mathbf{y}_1, \mathbf{y}_2) = q_1(\mathbf{y}_1)q_2(\mathbf{y}_2)$ *and is given by* $q_i(y) \propto \sqrt{p_1(\mathbf{y})p_2(\mathbf{y})}$.

Proof Taking the dual of the optimization problem in Equation 21 we get

$$\arg\max_{\lambda} -\log \sum_{\mathbf{y}_1, \mathbf{y}_2} p(\mathbf{y}_1, \mathbf{y}_2) \exp(\lambda \cdot \phi)$$

with $q(\mathbf{y}_1, \mathbf{y}_2) \propto p(\mathbf{y}_1, \mathbf{y}_2) \exp(\lambda \cdot \phi(\mathbf{y}_1, \mathbf{y}_2))$. Where $\phi(\mathbf{y}_1, \mathbf{y}_2)$ is a vector of features of the form $\delta(\mathbf{y}_1 = \mathbf{y}) - \delta(\mathbf{y}_2 = \mathbf{y})$ with one entry for each possible label $\mathbf{y}$. Noting that the features decompose into $\phi'(\mathbf{y}_1) - \phi'(\mathbf{y}_2)$, we know that $q(\mathbf{y}_1, \mathbf{y}_2)$ decomposes as $q_1(\mathbf{y}_1)q_2(\mathbf{y}_2)$. Furthermore, our constraints require that $q_1(\mathbf{y}) = q_2(\mathbf{y}) \forall \mathbf{y}$ so we have $q(\mathbf{y}_1)q(\mathbf{y}_2) \propto p_1(\mathbf{y}_1) \exp(\lambda \cdot \phi'(\mathbf{y}_1)) p_2(\mathbf{y}_2) \exp(-\lambda \cdot \phi'(\mathbf{y}_2))$. Letting $\mathbf{y}_1 = \mathbf{y}_2$ we have $q(\mathbf{y})^2 = p_1(\mathbf{y})p_2(\mathbf{y})$ which gives us a closed form computation of agree$(p_1, p_2) \propto \sqrt{p_1(\mathbf{y})p_2(\mathbf{y})}$. Substituting this solution into the problem of Proposition 6.1, and performing algebraic simplification yields the desired result. ■

Replacing the minimum KL term in Equation 2 with a Bhattacharyya regularization term yields the objective

$$\min_{\theta} \mathcal{L}_1(\theta_1) + \mathcal{L}_2(\theta_2) + c\mathbf{E}_U[B(p_1(\theta_1), p_2(\theta_2))]$$

where $\mathcal{L}_i = \mathbf{E}[-\log(p_i(\mathbf{y}_i|\mathbf{x}; \theta_i))] + \frac{1}{\sigma_i^2}||\theta_i||^2$ for $i = 1, 2$ are the standard regularized log likelihood losses of the models p_1 and p_2, $\mathbf{E}_U[B(p_1, p_2)]$ is the expected Bhattacharyya distance (Kailath, 1967) between the predictions of the two models on the unlabeled data, and c is a constant defining the relative weight of the unlabeled data relative to the labeled data.

Our regularizer extends to full agreement for undirected graphical models. In the case where p_1 and p_2 have the same structure, $q = \text{agree}(p_1, p_2)$ will share this structure and the projection can be computed in closed form.

Proposition 6.2 *Suppose $p_i(\mathbf{Y}|\mathbf{X}), i \in \{1,2\}$ factor as a set of clique potentials from a set of cliques $\mathcal{C}$:*

$$p_i(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z_i(\mathbf{X})} \prod_{c \in \mathcal{C}} \psi_i(\mathbf{X}, \mathbf{Y}_c),$$

then $q_i(\mathbf{Y})$ also factors as a product over clique potentials in $\mathcal{C}$, and can be computed in closed form modulo normalization as $q(\mathbf{Y}_1, \mathbf{Y}_2) = q_1(\mathbf{Y}_1) q_2(\mathbf{Y}_2)$ with

$$q_i(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z'(\mathbf{X})} \prod_{c \in \mathcal{C}} \sqrt{\psi_1(\mathbf{X}, \mathbf{Y}_c) \psi_2(\mathbf{X}, \mathbf{Y}_c)}.$$

Proof The proof is simple algebraic manipulation. We start with Equation 22, an application of Proposition 6.1.

$$\begin{aligned} q_i(\mathbf{Y})^2 &\propto p_1(\mathbf{Y}|\mathbf{X}) p_2(\mathbf{Y}|\mathbf{X}) \\ &= Z_1^{-1} Z_2^{-1} \prod_c \psi_1(\mathbf{X}, \mathbf{Y}_c) \psi_2(\mathbf{X}, \mathbf{Y}_c) \\ &= \left(\frac{1}{Z'(\mathbf{X})} \prod_c \sqrt{\psi_1(\mathbf{X}, \mathbf{Y}_c) \psi_2(\mathbf{X}, \mathbf{Y}_c)} \right)^2. \end{aligned} \tag{22}$$

■

Note that Proposition 6.2 is not a special case of Proposition 2.2 because we have defined one constraint feature $\phi_{\mathbf{y}}$ for each possible labeling $\mathbf{y}$, and these do not decompose according to $\mathcal{C}$. We could alternatively have proven that ensuring agreement on clique potentials is identical to ensuring agreement on labelings. In the case of log-linear Markov random fields, the clique potentials are stored in log space so computing q corresponds to averaging the values before computing normalization.

6.2 Partial Agreement and Hierarchical Labels

Our method extends naturally to partial-agreement scenarios. For example we can encourage two part-of-speech taggers with different tag sets to produce compatible parts of speech, such as noun in tag set one and singular-noun in tag set 2, as opposed to noun in tag set 1 and verb in tag set 2. In particular, suppose we have a mapping from both label sets into a common space where it makes sense to encourage agreement. For the part-of-speech tagging example, this could mean mapping all nouns from both tag sets into a single class, all verbs into another class and so on. In general suppose we have functions $g_1(\mathbf{y}_1)$ and $g_2(\mathbf{y}_2)$ that map variables for the two models onto the same space $\{\mathbf{z}\}$. Then, $p_i(\mathbf{y}_i)$ and g_i induce a distribution:

$$p_i(\mathbf{z}) = \sum_{\mathbf{y} \,:\, g_i(\mathbf{y}) = \mathbf{z}} p_i(\mathbf{y}) \quad \text{and} \quad p_i(\mathbf{y}_i|\mathbf{z}) = p_i(\mathbf{y}_i)/p_i(\mathbf{z}).$$

We can encourage $p_1(\mathbf{z}) \approx p_2(\mathbf{z})$ by adding a feature for each label in the joint space:

$$\phi_{\mathbf{z}}(\mathbf{y}_i) = \begin{cases} 1 & \text{if } i = 1 \text{ and } g_1(\mathbf{y}_1) = \mathbf{z} \\ -1 & \text{if } i = 2 \text{ and } g_2(\mathbf{y}_2) = \mathbf{z} \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

In this case our objective becomes:

$$\min_{\theta} \mathcal{L}_1(\theta_1) + \mathcal{L}_2(\theta_2) + c\mathbf{E}_U[B(p_1(\mathbf{z}), p_2(\mathbf{z}))] \ .$$

In the special case where some labels are identical for the two models and others are incompatible, we have $g_1(\mathbf{z}_1)$ mapping the incompatible labels into one bin and the others into their own special bins. Proposition 6.3 along with the optimization algorithm described in Section 2.6 allows us to optimize this objective.

Proposition 6.3 *The Bhattacharyya distance* $-\log \sum_{\mathbf{z}} \sqrt{p_1(\mathbf{z})p_2(\mathbf{z})}$ *is* $\frac{1}{2}$ *the value of the convex optimization problem*

$$\begin{aligned} \min_{q} \quad & \mathbf{KL}(q(\mathbf{Y}_1, \mathbf{Y}_2) || p_1(\mathbf{Y}_1)p_2(\mathbf{Y}_2)) \\ \text{s.t.} \quad & \mathbf{E}_q(\phi) = 0, \end{aligned}$$

where the constraint features ϕ *are defined as in Equation 23. Furthermore, the minimizer decomposes as* $q(\mathbf{Y}_1, \mathbf{Y}_2) = q_1(\mathbf{Y}_1|\mathbf{z}_1)q_1(\mathbf{z}_1)q_2(\mathbf{Y}_2|\mathbf{z}_2)q_2(\mathbf{z}_2)$*, where* $q_1(\mathbf{z}_1) = q_2(\mathbf{z}_2) \propto \sqrt{p_1(\mathbf{z}_1)p_2(\mathbf{z}_2)}$ *and* $q_i(\mathbf{Y}_i|\mathbf{z}_i) = p_i(\mathbf{Y}_i|\mathbf{z}_i)$ $i \in \{1,2\}$.

Note that the computation of agree(p_1, p_2) is still in closed form if our models are unstructured.

Unfortunately, if we collapse some labels for structured models, $p(\mathbf{Y})$ might not have the same Markov properties as $p(\mathbf{z})$. For example, consider the case where p is a distribution over three states (1,2,3) that assigns probability 1 to the sequence (1,2,3,1,2,3,...) and probability zero to other sequences. This is a first-order Markov chain. If the mapping is $1 \mapsto 1$ and $2,3 \mapsto 0$ then $p(\mathbf{y})$ assigns probability 1 to (1,0,0,1,0,0,...), which cannot be represented as a first-order Markov chain. Essentially, the original chain relied on being able to distinguish between the allowable transition (2,3) and the disallowed transition (3,2). When we collapse the states, both of these transitions map to (0,0) and cannot be distinguished. Consequently, the closed form solution given in Proposition 6.3 is not usable. Potentially, we could compute some approximation to $p(\mathbf{y})$ and from that compute an approximation to q. Instead, we re-formulate our constraints to require only that the marginals of each clique in p_1 and p_2 match each other rather than requiring the joint to have the same probability:

$$\phi_{c,\mathbf{z}_c}(\mathbf{Y}_1, \mathbf{Y}_2) = \begin{cases} 1 & \text{if } i = 1 \text{ and } g_1(\mathbf{y}_1)_c = \mathbf{z}_c \\ -1 & \text{if } i = 2 \text{ and } g_2(\mathbf{y}_2)_c = \mathbf{z}_c \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

By Proposition 2.2, the features in Equation 24 lead to a q that respects the Markov properties of the original models.

6.3 Relation to Other Multi-View Learning

To avoid a long detour from PR, we describe here only CoBoosting (Collins and Singer, 1999) and two view Perceptron (Brefeld et al., 2005), the frameworks with which we empirically compare our method in the next section. Since these methods are based on different objective functions from ours it is worth examining where each one works best. Altun et al. (2003) compare log-loss and exp-loss for sequential problems. They find that the loss function does not have as great an effect on performance as the feature choice. However, they also note that exp-loss is expected to

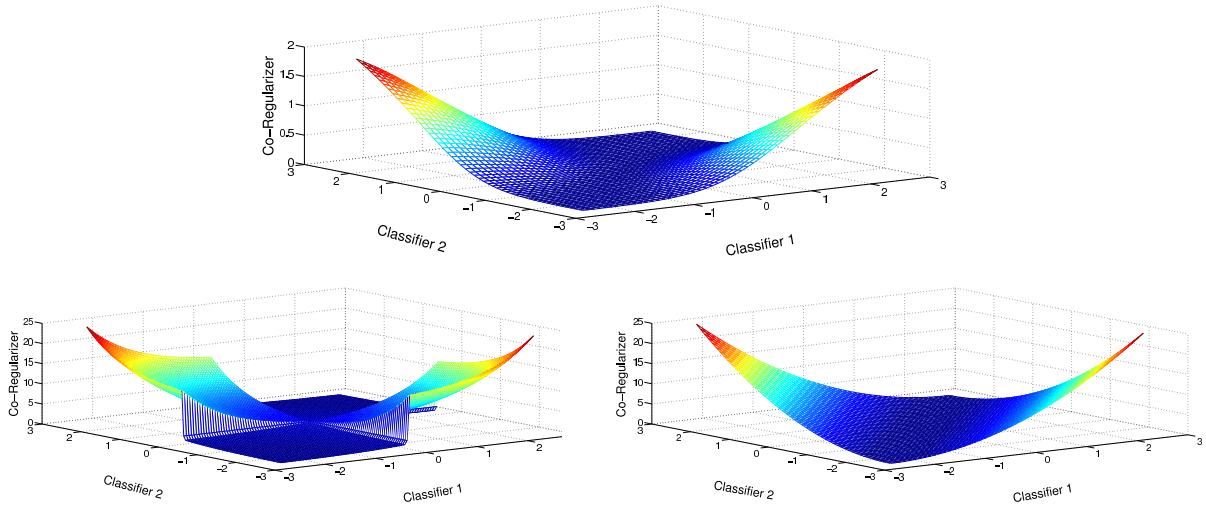


Figure 7: Different Loss Functions. **Top**: Bhattacharyya distance regularization. **Bottom left**: Exp-loss regularization. **Bottom right**: Least squares regularization.

perform better for clean data, while log-loss is expected to perform better when there is label noise. The intuition behind this is in the rate of growth of the loss functions. Exp-loss grows exponentially with misclassification margin while log-loss grows linearly. Consequently when there is label noise, AdaBoost focuses more on modeling the noise. Since CoBoosting optimizes a co-regularized exp-loss while our work optimizes a co-regularized log-loss we expect to do better on problems where the labels are noisy.

To get more intuition about this, Figure 7 shows the co-regularization loss functions for our method, CoBoosting, and co-regularized least squares (Sindhwani et al., 2005). For two underlying binary linear classifiers, $\hat{y}_1 = \text{sign}(w_1 \cdot x)$ and $\hat{y}_2 = \text{sign}(w_2 \cdot x)$, the horizontal axes represent the values $\hat{y}_1$ and $\hat{y}_2$, while the vertical axis is the loss. If we consider the plane parallel to the page, we see how the different co-regularizers penalize the classifiers when they disagree and are equally confident in their decision. Restricted to this plane, all three co-regularizers grow at the same asymptotic rate as the loss functions for the individual models: Linearly for our work, exponentially for CoBoosting and quadratically for co-RLS. If we look at the area where the two models agree (the flat part of the CoBoosting graph) we see what the penalty is when the classifiers agree but have different confidence. In this case co-RLS is harshest since it penalizes differences in the dot product equally regardless of the absolute value of the dot product. Intuitively, this is a problem. If one model predicts 1 with confidence 0.5 and the other predicts -1 with confidence 0.5 they are disagreeing while if they both predict 1 with confidence 1000 and 1001 respectively, they are agreeing on the label and are very close in their confidence estimates. At the other extreme, CoBoosting imposes almost no penalty whenever the two classifiers agree, regardless of their confidence. The Bhattacharyya distance co-regularizer lies between these extremes, penalizing differences in confidence near the origin but is more lenient when the classifiers are both very confident and agree.

Finally, if we have labeled data from one domain but want to apply it to another domain we can use any of the co-training frameworks mentioned earlier, including our own, to perform do-

Domains	MIRA	Boost	Perc	mx-ent	SCL	CoBoost	coPerc	PR
books→dvds	77.2	72.0	74	78.5	75.8	78.8	75.5	**79.8**
dvds→books	72.8	74.8	74.5	80.3	79.7	79.8	74.5	**81.3**
books→electr	70.8	70.3	73.3	72.5	75.9	**77.0**	69.3	75.5
electr→books	70.7	62.5	73	72.8	**75.4**	71.0	67.5	74.3
books→kitchn	74.5	76.3	73.5	77.8	78.9	78.0	76.5	**81.0**
kitchn→books	70.9	66.5	67.3	70.3	68.6	69.8	66	**72.8**
dvds→electr	73.0	73.2	73.5	75.5	74.1	75.3	71.2	**76.5**
electr→dvds	70.6	66.3	64.8	69.3	**76.2**	73.5	63.3	73.0
dvds→kitchn	74.0	75.5	78.3	80.5	81.4	79.0	78.25	**82.8**
kitchn→dvds	72.7	61.8	64	69.5	**76.9**	70.1	60.5	72.8
electr→kitchn	84.0	73.2	81	**86.5**	85.9	85.0	83.3	85.8
kitchn→electr	82.7	66.3	81	82.8	**86.8**	83.0	80.5	85.5
Average improvement	-1.87	-6.47	-3.18	N/A	1.61	0.33	-4.16	**2.07**
Standard deviation	3.04	4.73	2.21	N/A	3.31	2.03	2.12	**1.27**

Table 4: Performance of several methods on a sentiment classification transfer learning task. Reviews of objects of one type are used to train a classifier for reviews of objects of another type. The abbreviations in the column names are as follows. Boost: AdaBoost algorithm, Perc: Perceptron, mx-ent: maximum entropy, SCL: structural correspondence learning, CoBoost: CoBoosting, coPerc: two view Perceptron, PR: this work. The best accuracy is shown in bold for each task. The last two rows of the table show the average improvement over maximum entropy (the best performing supervised method), and also the standard deviation of the improvement.

main transfer. For sentiment classification we will see that our method performs comparably with Structural Correspondence Learning (Blitzer et al., 2006), which is based on Alternating Structure Optimization (Ando and Zhang, 2005).

6.4 Experiments

Our first set of experiments is for transfer learning for sentiment classification. We use the data from Blitzer et al. (2007). The two views are generated from a random split of the features. We compare our method to several supervised methods as well as CoBoosting (Collins and Singer, 1999), two view Perceptron (Brefeld et al., 2005) and structural correspondence learning (Blitzer et al., 2007). Results are in Table 4. The column labeled "SCL" contains the best results from Blitzer et al. (2007), and is not directly comparable with the other methods since it uses some extra knowledge about the transfer task to choose auxiliary problems. For all the two-view methods we weigh the total labeled data equally with the total unlabeled data. We regularize the maximum entropy classifiers with a unit variance Gaussian prior. Out of the 12 transfer learning tasks, our method performs best in 6 cases, SCL in 4, while CoBoosting performs best only once. Two view Perceptron never outperforms all other methods. One important reason for the success of our method is the relative strength of the maximum entropy classifier relative to the other supervised methods for this

particular task. We expect that CoBoosting will perform better than our method in situations where Boosting significantly out-performs maximum entropy.

The next set of our experiments are on named entity disambiguation. Given a set of already segmented named entities, we want to predict what type of named entity each one is. We use the training data from the 2003 CoNLL shared task (Sang and Meulder, 2003). The two views comprise content versus context features. The content features are words, POS tags and character n-grams of length 3 for all tokens in the named entity, while context features the same but for three words before and after the named entity. We used 2000 examples as test data and roughly 30,000 as unlabeled (train) data. Table 5 shows the results for different amounts of labeled train data. For this data, we choose the variance of the Gaussian prior as well as the relative weighting of the labeled and unlabeled data by cross validation on the train set. In order to test whether the advantage our method gets is from the joint objective or from the use of agree(p_1, p_2), which is an instance of logarithmic opinion pools, we also report the performance of using agree(p_1, p_2) when the two views p_1 and p_2 have been trained only on the labeled data. In the column labeled "agree$_0$" we see that for this data set the benefit of our method comes from the joint objective function rather than from the use of logarithmic opinion pools.

Data size	mx-ent	agree$_0$	PR	RRE
500	74.0	74.4	76.4	9.2%
1000	80.0	80.0	81.7	8.5%
2000	83.4	83.4	84.8	8.4%

Table 5: Named entity disambiguation. Prior variance and c chosen by cross validation. agree$_0$ refers to performance of two view model before first iteration of EM. RRE is reduction in error relative to error of MaxEnt model.

In order to investigate the applicability of our method to structured learning we apply it to the shallow parsing task of noun phrase chunking. We our experiments are on the English training portion of the CoNLL 2000 shared task (Sang and Buchholz, 2000). We select 500 sentences as test data and varying amounts of data for training; the remainder was used as unlabeled (train) data. We use content and context views, where the content view is the current word and POS tag while the context view is the previous and next words and POS tags. We regularize the CRFs with a variance 10 Gaussian prior and weigh the unlabeled data so that it has the same total weight as the labeled data. The variance value was chosen based on preliminary experiments with the data. Table 6 shows the F-1 scores of the different models. We compare our method to a monolithic CRF as well as averaged Perceptron the two view Perceptron of Brefeld et al. (2005) with averaging. The Perceptron models were trained for 20 iterations. Preliminary experiments show that performance on held out data does not change after 10 iterations so we believe the models have converged. Both two view semi-supervised methods show gains over the corresponding fully-supervised method for 10-100 sentences of training data, but do not improve further as the amount of labeled data increases. The method presented in this paper out-performs two view Perceptron when the amount of labeled data is very small, probably because regularized CRFs perform better than Perceptron for small amounts of data. As the number of training sentences increases, two view Perceptron performs as well as our method, but at this point it has little or no improvement over the fully-supervised Perceptron.

size	CRF	SAR(RRE)	Perc	coPerc
10	73.2	**78.2**(19%)	69.4	71.2
20	79.4	**84.2**(23%)	74.4	76.8
50	86.3	**86.9** (4%)	80.1	84.1
100	88.5	**88.9** (3%)	86.1	88.1
200	89.6	89.6 (0%)	89.3	**89.7**
500	**91.3**	90.6 (-8%)	90.8	90.9
1000	91.6	91.1 (-6%)	91.5	**91.8**

Table 6: F-1 scores for noun phrase chunking with context/content views. Test data comprises 500 sentences, with 8436 sentences divided among labeled and unlabeled train data. The best score is shown in bold for each train data size.

7. Cross Lingual Projection

For English and a handful of other languages, there are large, well-annotated corpora with a variety of linguistic information ranging from named entity to discourse structure. Unfortunately, for the vast majority of languages very few linguistic resources are available. This situation is likely to persist because of the expense of creating annotated corpora that require linguistic expertise (Abeillé, 2003). On the other hand, parallel corpora between many resource-poor languages and resource-rich languages are ample, motivating recent interest in transferring linguistic resources from one language to another via parallel text.

Dependency grammars are one such resource. They are useful for language modeling, textual entailment and machine translation (Haghighi et al., 2005; Chelba et al., 1997; Quirk et al., 2005; Shen et al., 2008), to name a few tasks. Dependency grammars are arguably more robust to transfer than constituent grammars, since syntactic relations between aligned words of parallel sentences are better conserved in translation than phrase structure (Fox, 2002; Hwa et al., 2005). The two main challenges to accurate training and evaluation from aligned bitext are: (1) errors in word alignments and source language parses, (2) unaligned words due to non-literal or distant translation.

Hwa et al. (2005) proposed to learn generative dependency grammars using Collins' parser (Collins, 1999) by constructing full target parses via projected dependencies. To address challenge (1), they introduced on the order of one to two dozen language-specific transformation rules. To address challenge (2), they used a set of tree-completion rules. We present here an alternative approach to dependency grammar transfer. Our approach uses a single, intuitive PR constraint to guide grammar learning. With this constraint, we avoid the need for complex tree completion rules and many language-specific rules, yet still achieve acceptable parsing accuracy.

It should be noted that while our source of supervision, a bitext, is the same as that of Hwa et al. (2005), our learning method is more closely related to that of Druck et al. (2009). They use the GE framework to train a dependency parser. Their source of supervision comes in the form of corpus-wide expected values of linguistic rules provided by a linguistic informant.

In what follows, X will indicate parallel part-of-speech tagged sentences in a bitext corpus, along with a dependency parse of the source language. Y will indicate the dependency parses for the target language sentences.

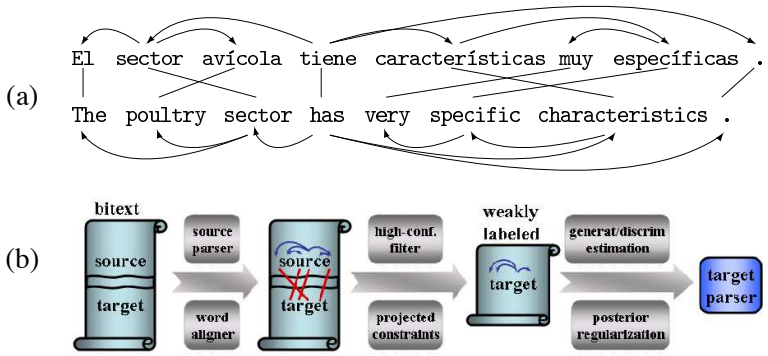


Figure 8: (a) An example word-aligned sentence pair with perfectly projected dependencies. (All dependency edges are conserved in this case.) (b) Overview of our grammar induction approach via bitext: the source (English) is parsed and word-aligned with target; after filtering, projected dependencies define constraints over target parse tree space, providing weak supervision for learning a target grammar.

7.1 Approach

Figure 8(a) shows an aligned sentence pair example where dependencies are perfectly "conserved" across the alignment. An edge from English parent p to child c is called conserved if word p aligns to word p' in the second language, c aligns to c' in the second language, and p' is the parent of c'. Note that we are not restricting ourselves to one-to-one alignments here; p, c, p', and c' can all also align to other words. Unfortunately the sentence in Figure 8(a) is highly unusual in its amount of dependency conservation, so we need to do more than directly transfer conserved edges to get good parsing accuracy.

The key to our approach is a single PR constraint, which ensures that the expected proportion of conserved edges in a sentence pair is at least η (the exact proportion we used was 0.9, which was determined using unlabeled data as described in the experiments section). Specifically, let $C_{\mathbf{x}}$ be the set of directed edges projected from English for a given sentence $\mathbf{x}$. Then given a parse $\mathbf{y}$, the proportion of conserved edges is $\phi(\mathbf{x},\mathbf{y}) = \frac{1}{|C_{\mathbf{x}}|}\sum_{y\in\mathbf{y}} \mathbf{1}(y \in C_{\mathbf{x}})$ and the expected proportion of conserved edges under distribution $p(\mathbf{y} \mid \mathbf{x})$ is

$$\mathbf{E}_p[\phi(\mathbf{x},\mathbf{y})] = \frac{1}{|C_{\mathbf{x}}|}\sum_{y\in C_{\mathbf{x}}} p(y \mid \mathbf{x}).$$

Consider how this constraint addresses errors in word alignment and source language parses, challenge (1) from above. First, note that we are constraining groups of edges rather than a single edge. For example, in some sentence pair we might find 10 edges that have both end points aligned and can be transferred. Rather than requiring our target language parse to contain each of the 10 edges, we require that the expected number of edges from this set is at least 10η. This gives the parser freedom to have some uncertainty about which edges to include, or alternatively to choose to exclude some of the transferred edges.

Our constraint does not address unaligned words due to non-literal or distant translation, challenge (2), as directly. Yet, we find that the constraint sufficiently limits the distribution over possible parses of unaligned words, such that the parser still makes reasonable choices for them. It is also

Basic Uni-gram Features

x_i-word, x_i-pos
x_i-word
x_i-pos
x_j-word, x_j-pos
x_j-word
x_j-pos

Basic Bi-gram Features

x_i-word, x_i-pos, x_j-word, x_j-pos
x_i-pos, x_j-word, x_j-pos
x_i-word, x_j-word, x_j-pos
x_i-word, x_i-pos, x_j-pos
x_i-word, x_i-pos, x_j-word
x_i-word, x_j-word
x_i-pos, x_j-pos

In Between POS Features

x_i-pos, b-pos, x_j-pos

Surrounding Word POS Features

x_i-pos, x_i-pos+1, x_j-pos-1, x_j-pos
x_i-pos-1, x_i-pos, x_j-pos-1, x_j-pos
x_i-pos, x_i-pos+1, x_j-pos, x_j-pos+1
x_i-pos-1, x_i-pos, x_j-pos, x_j-pos+1

Table 7: Features used by the MSTParser. For each edge (i, j), x_i-word is the parent word and x_j-word is the child word, analogously for POS tags. The +1 and -1 denote preceeding and following tokens in the sentence, while b denotes tokens between x_i and x_j.

worth noting that if we wished to more directly address challenge (2), we could add additional constraint features to the PR framework. For example, it seems intuitive that unaligned words might tend to be leaves (e.g., articles that are dropped in some languages but not in others). Thus, one constraint we could enforce would be to restrict the number of children of unaligned words to fall below some threshold.

For both models, we compute the projection onto the constraint set using a simple line search. This is possible since there is only one constraint per sentence and the constraints do not interact.

At a high level our approach is illustrated in Figure 8(b). A parallel corpus is word-level aligned using the method described in Section 5, where we train an alignment model via PR using symmetry constraints. The source (English) is parsed using a dependency parser (McDonald et al., 2005). Then, the filtering stage eliminates low-confidence alignments such as noun-to-verb alignments, restricts the training set to exclude possible sentence fragments, and follows the method of Klein and Manning (2004) in stripping out punctuation. We then learn a probabilistic parsing model using PR. In our experiments we evaluate the learned models on dependency treebanks (Nivre et al., 2007).

7.2 Parsing Models

We explored two parsing models: a generative model used by several authors for unsupervised induction and a discriminative model previously used for fully supervised training.

The discriminative parser is based on the edge-factored model and features of the MSTParser (McDonald et al., 2005). The parsing model defines a conditional distribution $p_\theta(\mathbf{y} \mid \mathbf{x})$ over each projective parse tree $\mathbf{y}$ for a particular sentence $\mathbf{x}$, parameterized by a vector θ. The probability of any particular parse is

$$p_\theta(\mathbf{y} \mid \mathbf{x}) \propto \prod_{y \in \mathbf{y}} e^{\theta \cdot \mathbf{f}(y, \mathbf{x})},$$

where y is a directed edge contained in the parse tree $\mathbf{y}$ and $\mathbf{f}$ is a feature function. In the fully supervised experiments we run for comparison, parameter estimation is performed by stochastic gradient ascent on the conditional likelihood function, similar to maximum entropy models or conditional random fields. One needs to be able to compute expectations of the features $\mathbf{f}(y, \mathbf{x})$ under the distribution $p_\theta(y \mid \mathbf{x})$. A version of the inside-outside algorithm (Lee and Choi, 1997) performs this computation. Viterbi decoding is done using Eisner's algorithm (Eisner, 1996).

We also used a generative model based on dependency model with valence (Klein and Manning, 2004). Under this model, the probability of a particular parse $\mathbf{y}$ and a sentence with part-of-speech tags $\mathbf{x}$ is given by

$$p_\theta(\mathbf{y},\mathbf{x}) = p_{\text{root}}(r(\mathbf{x})) \cdot \Big(\prod_{y\in\mathbf{y}} p_{\neg\text{stop}}(y_p,y_d,v_y)\, p_{\text{child}}(y_p,y_d,y_c)\Big) \cdot \Big(\prod_{x\in\mathbf{x}} p_{\text{stop}}(x,\text{left},v_l)\, p_{\text{stop}}(x,\text{right},v_r)\Big)$$

where $r(\mathbf{x})$ is the part-of-speech tag of the root of the parse tree $\mathbf{y}$, y is an edge from parent y_p to child y_c in direction y_d, either left or right, and v_y indicates valency—false if y_p has no other children further from it in direction y_d than y_c, true otherwise. The valencies v_r/v_l are marked as true if x has any children on the left/right in $\mathbf{y}$, false otherwise.

We regularize the models by parameter prior $-\log p(\theta) = R(\theta)$, where $p(\theta)$ is Gaussian for the discriminative model and Dirichlet for the generative.

7.3 Experiments

We evaluate our approach by transferring from an English parser trained on the Penn treebank to Bulgarian and Spanish. We evaluate our results on the Bulgarian and Spanish corpora from the CoNLL X shared task. The Bulgarian experiments transfer a parser from English to Bulgarian, using the OpenSubtitles corpus (Tiedemann, 2007). The Spanish experiments transfer from English to Spanish using the Spanish portion of the Europarl corpus (Koehn, 2005). For both corpora, we performed word alignments with the open source PostCAT (Graça et al., 2009b) toolkit. We used the Tokyo tagger (Tsuruoka and Tsujii, 2005) to POS tag the English tokens, and generated parses using the first-order model of McDonald et al. (2005) with projective decoding, trained on sections 2-21 of the Penn treebank with dependencies extracted using the head rules of Yamada and Matsumoto (2003). For Bulgarian we trained the Stanford POS tagger (Toutanova et al., 2003) on the Bulgtreebank corpus from CoNLL X. The Spanish Europarl data was POS tagged with the FreeLing language analyzer (Atserias et al., 2006). The discriminative model used the same features as MSTParser, summarized in Table 7. Our model uses constraints of the form: the expected proportion of conserved edges in a sentence pair is at least $\eta = 90\%$.[10]

In order to better evaluate our method, we construct a baseline inspired by Hwa et al. (2005). The baseline creates a full parse tree from the incomplete and possibly conflicting transferred edges using a simple random process. We start with no edges and try to add edges one at a time verifying at each step that it is possible to complete the tree. We first try to add the transferred edges in random order, then for each orphan node we try all possible parents (both in random order). We then use this full labeling as supervision for a parser. Note that this baseline is very similar to the first iteration of our model, since for a large corpus the different random choices made in different sentences tend to smooth each other out. We also tried to create rules for the adoption of orphans, but the simple rules we tried added bias and performed worse than the baseline we report.

10. We chose η in the following way: We split the unlabeled parallel text into two portions. We trained models with different η on one portion and ran it on the other portion. We chose the model with the highest fraction of conserved constraints on the second portion.

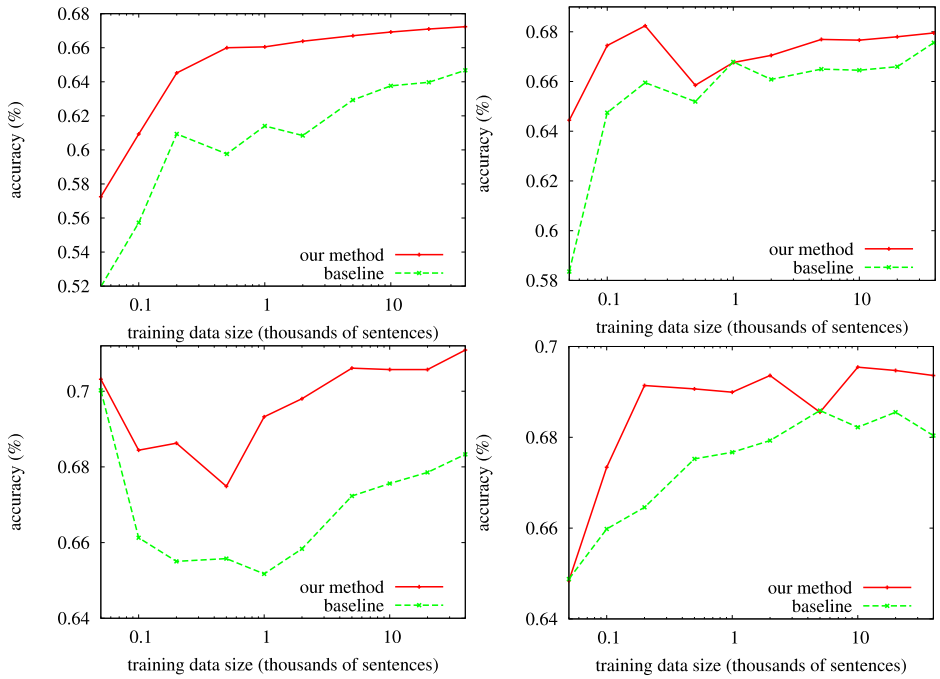


Figure 9: Learning curves. Each graph compares transferring a single tree of edges (baseline) and transferring all possible projected edges (our method). The models were trained on sentences of length up to 20 and tested on CoNLL train sentences of length up to 10. Punctuation was stripped at train time. **Top**: Bulgarian. **Bottom**: Spanish. **Left**: Discriminative model. **Right**: Generative model. The non-monotonicity of the (such as bottom left) is because each point is based on a single random sample of sentences. This random selection can greatly affect performance when the number of sentences is small.

7.3.1 RESULTS

Models are evaluated based on attachment accuracy—the fraction of words assigned the correct parent. Figure 9 shows that models generally improve with more transfer-type data. It also shows our method consistently outperforms the baseline. Note that each point in these graphs is based on a single random subsample of the data, which leads to some non-monotonicity in the left-half of some of the curves. The exact accuracy numbers for the 10k training sentences point of Figure 9 are given in Table 8. Link-left baselines for these corpora are much lower: 33.8% and 27.9% for Bulgarian and Spanish respectively.

7.3.2 GENERATIVE PARSER

The generative model we use is a state of the art model for unsupervised parsing. Before evaluating, we smooth the resulting models by adding e^{-10} to each learned parameter, merely to remove the chance of zero probabilities for unseen events. (We did not bother to tune this value at all as it makes very little difference for final parses.) Unfortunately, we found generative model performance was disappointing in the unsupervised setting. Using the initialization procedure from Klein and Man-

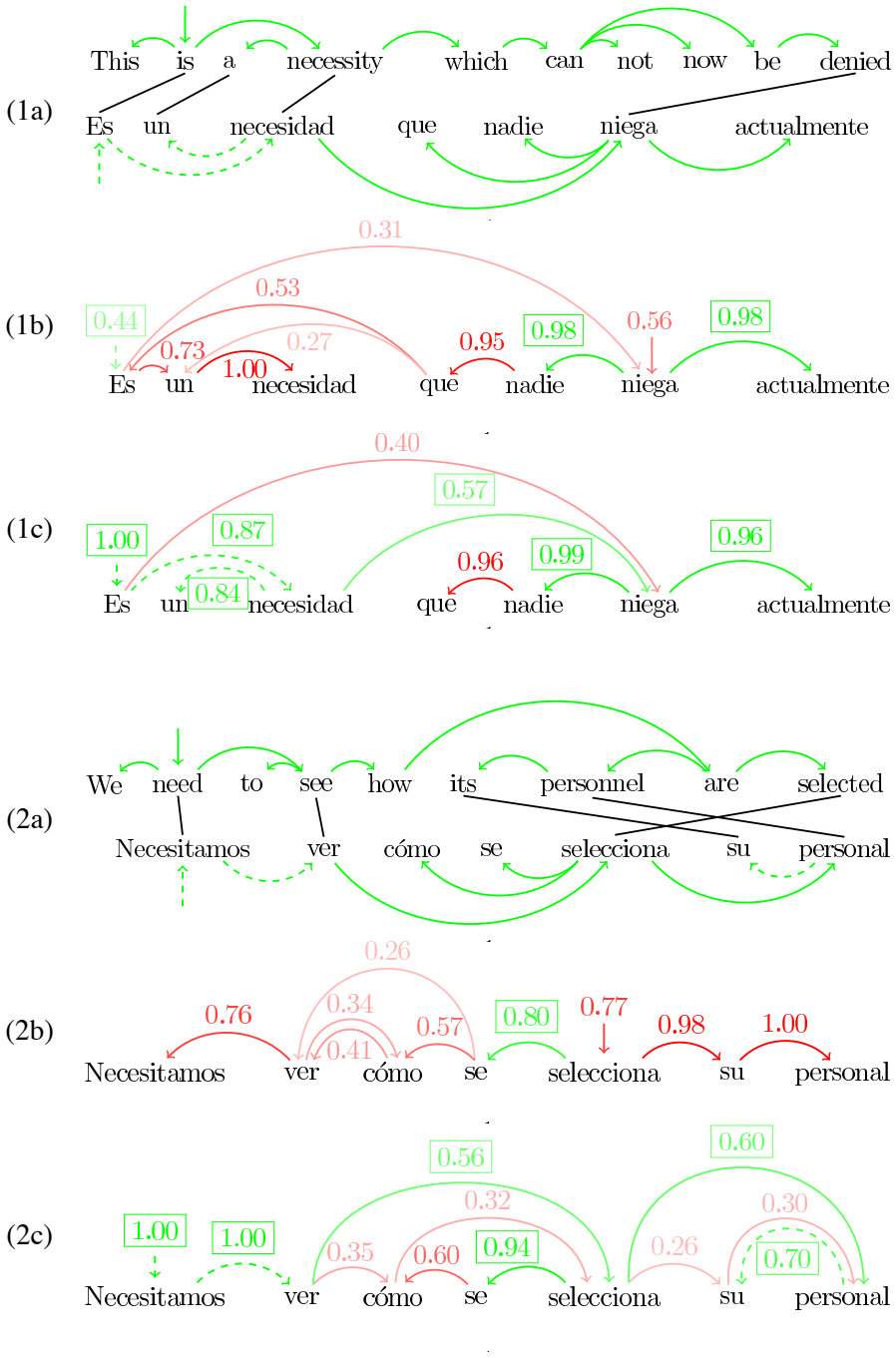


Figure 10: Posteriors of two Spanish sentences from Europarl. The number on each edge indicates the edge's posterior probability. Edges with probability less than 0.25 are not shown. Darker (more saturated) edges are higher probability. Green (with boxed number) indicates a correct edge, red (no box) an incorrect. Dotted edges are conserved. **(a)** The gold source and target parses and their alignment. **(b)** Unsupervised model initialized as per Klein and Manning (2004) and trained for 100 EM iterations. **(c)** PR projection applied to the posteriors of the middle figure, forcing $\mathbf{E}_p[C] \geq C * \eta = 3 * 0.9$, where C is the number of conserved edges.

	Discriminative		Generative	
	Bulgarian	Spanish	Bulgarian	Spanish
Baseline	63.8	67.6	66.5	68.2
Post.Reg.	**66.9**	**70.6**	**67.8**	**69.5**

Table 8: Accuracy values at the 10k training sentences point of Figure 9.

ning (2004), the maximum unsupervised accuracy it achieves is 55.4% for Bulgarian and 41.7% for Spanish, and these results are not stable. Changing the initialization parameters or training sample drastically affects the results, even for samples with several thousand sentences. But when we use the transferred information to constrain the learning, EM stabilizes and achieves much better performance, also beating the Hwa et al. (2005)-inspired baseline. With the transferred information, even setting all parameters equal at the outset does not prevent the model from learning the dependency structure of the aligned language. Figure 10 shows an example of how PR projection helps better estimate posteriors of two example sentences.

7.3.3 DISCRIMINATIVE PARSER

We trained our discriminative parser for 100 iterations of online EM with a Gaussian prior variance of 100. The transfer system performs better than the unsupervised generative model and the baseline model for both Bulgarian and Spanish. We observed another desirable property of the discriminative model: While the generative model can get confused and perform poorly when the training data contains very long sentences, the discriminative parser does not appear to have this drawback. In fact we observed that as the maximum training sentence length increased, the parsing performance also improved.

8. Enforcing Sparsity Structure

Many important NLP tasks (e.g., tagging, parsing, named-entity recognition) involve word classification. Often, we know a priori that a word type might belong to a small set of classes (where the class of a specific instance depends on its context) and should never belong to any of the many possible classes outside this small set. The part-of-speech tagging task, described in the running example, is one instance of this phenomenon. For example, consider the word type "run". It might belong to the verb class in some instances and the noun class in others, but it will never be an adjective, adverb, conjunction, determiner, etc. Learning algorithms typically assume that each word type can be associated with any existing tag, even though in reality each word type is only ever associated with a few tags.

Unsupervised induction of this latent structure is normally performed using the EM algorithm, but it has exhibited disappointing performance in previous work. One well-known reason for this is that EM tends to allow each word to be generated by most POS tags some of the time. In reality, we would like most words to have a small number of possible POS tags. Previous work has attempted to solve this problem by applying the Bayesian approach, using a prior to encourage sparsity in the model *parameters* (Gao and Johnson, 2008; Johnson, 2007; Goldwater and Griffiths, 2007). However, this approach has the drawback of enforcing sparsity in the wrong direction; sparsity at the parameter level encodes a preference that each POS tag should generate only a few words,

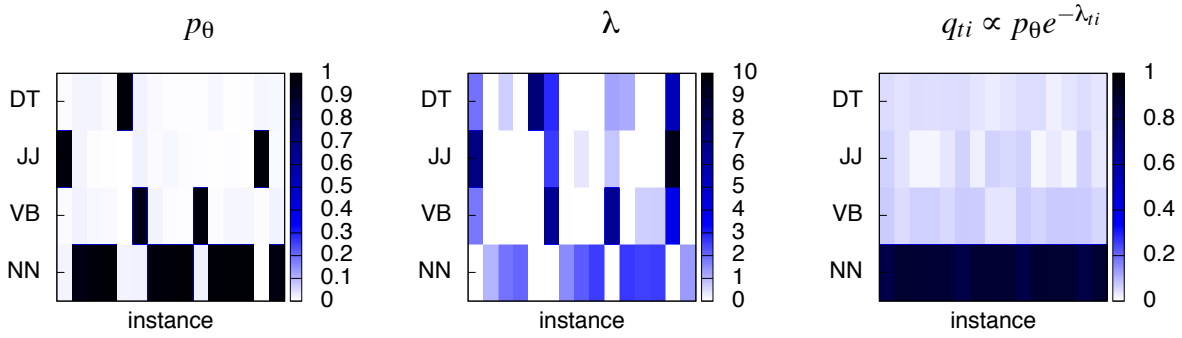


Figure 11: An illustration of ℓ_1/ℓ_∞ regularization. Left panel: initial tag distributions (columns) for 15 instances of a word. Middle panel: optimal regularization parameters λ, each row sums to $\sigma = 20$. Right panel: q concentrates the posteriors for all instances on the NN tag, reducing the ℓ_1/ℓ_∞ norm from just under 4 to a little over 1.

instead of encoding that each word should generate only a few POS tags. Here we explore the problem of biasing unsupervised models to favor the correct sparsity by encouraging the model to achieve *posterior* sparsity on unlabeled training data.

8.1 ℓ_1/ℓ_∞ Regularization

We focus on the slack-penalized formulation of Section 2.3 for this task. We choose the PR constraint to encourage each word to be associated with only a few parts of speech. Let the constraint feature $\phi_{wti}(\mathbf{X}, \mathbf{Y})$ have value 1 whenever the i^{th} occurrence of word w has part-of-speech tag t, and value 0 otherwise. For every word w, we would like there to be only a few POS tags t such that there are occurrences i where t has nonzero probability. This can be achieved if it "costs" a lot the first time an occurrence of a word takes a particular tag, but afterwards future occurrences of the word can receive that same tag for free. More precisely, for each word type w, we would like the sum (ℓ_1 norm), over tags t, of the maximum (ℓ_∞ norm), over all occurrences w_i of w, of $p(w_i \mid t)$, to be small; we want $\sum_{t,w} \max_i p(w_i \mid t)$ to be small. Note that in contrast to previous applications, these constraints are corpus-wide instead of instance-specific. For notational simplicity we will write $\phi_{wti}(\mathbf{X}, \mathbf{Y}) = \phi_{wti}(\mathbf{Y})$, since any dependence on $\mathbf{X}$ is captured in the subscripts of ϕ_{wti}.

Formally, this objective is an example of the slack-penalized formulation (as in Equation 3), but for simplicity we will use the notation:

$$\min_{q, c_{wt}} \ \mathbf{KL}(q||p_\theta) + \sigma \sum_{wt} c_{wt} \quad \text{s.t.} \quad \mathbf{E}_q[\phi_{wti}] \leq c_{wt}.$$

Mapping this notation to the original equation we have: $\mathbf{b} = 0$ and regularization strength σ. The constraints on the features ϕ_{wti} and the summation over c_{wt} together encode the ℓ_1/ℓ_∞ norm. The variables c_{wt} represent the ℓ_∞ norm of ϕ_{wti}, $c_{wt} = ||\phi_{wti}||_{\ell_\infty}$, while the summation is the ℓ_1 norm of c_{wt}. The dual of this objective has a very simple form:

$$\max_{\lambda \geq 0} \ -\log \left(\sum_y p_\theta(\mathbf{Y}) \exp(-\boldsymbol{\lambda} \cdot \phi(\mathbf{Y})) \right) \quad \text{s.t.} \quad \sum_t \lambda_{wti} \leq \sigma, \tag{25}$$

where $\mathbf{Y}$ ranges over assignments to the hidden tag variables for all of the occurrences in the training data, $\phi(\mathbf{Y})$ is the vector of ϕ_{wti} constraint feature values for assignment $\mathbf{Y}$, $\boldsymbol{\lambda}$ is the vector of dual parameters λ_{wti}, and the primal parameters are $q(\mathbf{Y}) \propto p_\theta(\mathbf{Y}) \exp(-\boldsymbol{\lambda} \cdot \phi(\mathbf{Y}))$.

An advantage of using slack penalties in this case is that ℓ_1/ℓ_∞ as a slack constraint in the primal would lead to a non-differentiable dual penalty term, which somewhat complicates optimization. Using a slack penalty makes sparsity regularization a primal penalty, yielding dual simplex constraints, solvable efficiently via projected gradient, as described by Bertsekas (1999). Note that the simplex constraints in Equation 25 can be interpreted as an ℓ_∞/ℓ_1 norm, which is the dual of the ℓ_1/ℓ_∞.

Figure 11 illustrates how the ℓ_1/ℓ_∞ norm operates on a toy example. For simplicity suppose we are only regularizing one word and our model p_θ is just a product distribution over 15 instances of the word. The left panel in Figure 11 shows the posteriors under p_θ. We would like to concentrate the posteriors on a small subset of rows. The center panel of the figure shows the $\boldsymbol{\lambda}$ values determined by Equation 25, and the right panel shows the projected distribution q, which concentrates most of the posterior on the bottom row. Note that we are not requiring the posteriors to be sparse, which would be equivalent to preferring that the distribution is peaked; rather, we want a word to concentrate its tag posterior on a few tags across all instances of the word. Indeed, most of the instances (columns) become less peaked than in the original posterior to allow posterior mass to be redistributed away from the outlier tags. Since they are more numerous than the outliers, they moved less. This also justifies only regularizing relatively frequent events in our model.

8.2 Results

In this section we present an empirical comparison of first-order HMMs trained with three different methods: classic EM (EM), ℓ_1/ℓ_∞ PR (Sparse), and Bayesian estimation using a variational approximation described in Johnson (2007) and Gao and Johnson (2008) (VEM). Models are trained and tested on three different corpora: the Wall Street Journal portion of the Penn treebank (Marcus et al., 1993) using a reduced set of 17 tags (Smith et al., 2005) (PTB17); the Bosque subset of the Portuguese Floresta Sinta(c)tica Treebank (Afonso et al., 2002)[11] used for the ConLL X shared task on dependency parsing (PT-CoNLL)[12]; and the Bulgarian BulTreeBank (Simov et al., 2002) (BulTree) with 12 coarse tags. All words that occurred only once were replaced by the token "unk". To measure model sparsity, we compute the average ℓ_1/ℓ_∞ norm over words occurring more than 10 times; the label 'L1LMax' denotes this measure in figures. Table 9 gives statistics for each corpus as well as the sparsity for a first-order HMM trained on the labeled data.

Following Gao and Johnson (2008), the parameters were initialized with a "pseudo E-step" as follows: we filled the expected count matrices with numbers $1 + X \times U(0,1)$, where $U(0,1)$ is a random number between 0 and 1 and X is a parameter. These matrices are then fed to the M-step; the resulting "random" transition and emission probabilities are used for the first real E step. For VEM X was set to 0.0001 (almost uniform) since this showed a significant improvement in performance. On the other hand EM showed less sensitivity to initialization, and we used $X = 1$ which resulted in the best results. The models were trained for 200 iterations as longer runs did not significantly change the results. For VEM we tested 4 different prior combinations based on the results of Johnson (2007); in later work Gao and Johnson (2008) considered a wider range of values

11. The subset can be found at `http://www.linguateca.pt/Floresta/`.
12. The task can be found at `http://nextens.uvt.nl/~conll/`.

	Types	Tokens	Unk	Tags	ℓ_1/ℓ_∞
PTB17	23768	950028	2%	17	1.23
PT-Conll	11293	206678	8.5%	22	1.14
BulTree	12177	174160	10%	12	1.04

Table 9: Corpus statistics. All words with only one occurrence where replaced by the 'unk' token. The third column shows the percentage of tokens replaced. ℓ_1/ℓ_∞ is the value of the sparsity for a fully supervised HMM trained in all available data.

but did not identify definitely better choices. Sparse was initialized with the parameters obtained by running EM for 30 iterations, followed by 170 iterations of the new training procedure. Predictions were obtained using posterior decoding since this consistently showed small improvements over Viterbi decoding.

We compare the models by measuring the mutual information between the distribution of hidden states and the distribution of the truth. Ideally, a perfect method would have mutual information equal to the entropy of both distributions. The farther the distribution that a method produces is from the truth the smaller the information gain is. We also evaluate the accuracy of the models using two established mappings between hidden states and POS tags: **(1-Many**) maps each hidden state to the tag with which it co-occurs the most; **1-1** (Haghighi and Klein, 2006) greedily picks a tag for each state under the constraint of never using the same tag twice. This results in an approximation of the optimal 1-1 mapping. If the numbers of hidden states and tags are not the same, some hidden states will be unassigned (and hence always wrong) or some tags not used. In all our experiments the number of hidden states is the same as the number of POS tags.

Figure 12 (Top Left) shows mutual information between the hidden state distribution of each method and the truth. The entropy of the true distribution are: BulTree 3.05, PT-CoNLL 3.49 and PTB17 3.22. Sparse is the method that achieves the biggest information gain across all corpora, and is not particularly sensitive to the strength of regularization used. Interestingly, VEM often has the smallest ℓ_1/ℓ_∞, even though mutual information is often worst than EM.

Figure 12 (Top Right) shows the different average values of the L1LMax statistics for each method across corpora. We see that both VEM and Sparse achieve values of ℓ_1/ℓ_∞ close to the gold standard, on the other hand EM as expected as bigger values which confirms the intuition that EM allows each word to be generated by most of the possible POS tags.

Figure 12 (Bottom Left) shows errors for all methods on the different corpora after 10 random initializations using the 1-Many mapping. For both VEM and Sparse we pick parameter settings resulting in the best average performance. A first conclusion is that using the ℓ_1/ℓ_∞ constraint consistently and significantly improves the results when compared with the other two methods.

Figure 12 (Bottom Right) shows the same errors for the 1-1 mapping. In this case Sparse still beats the EM but does not always outperform VEM. One reason for this behavior is that this metric is very sensitive to the number of word types associated with each hidden state. VEM tends to encourage some large hidden states with many word types, which is preferable using the 1-1 mapping for large word categories such as nouns. On the other hand Sparse tends to spread the nouns over 4 different hidden states. This difference is particularly pronounced for the condensed tag sets (PTB17, PT-CoNLL) where different kinds of nouns are joined into one large tag. Also this

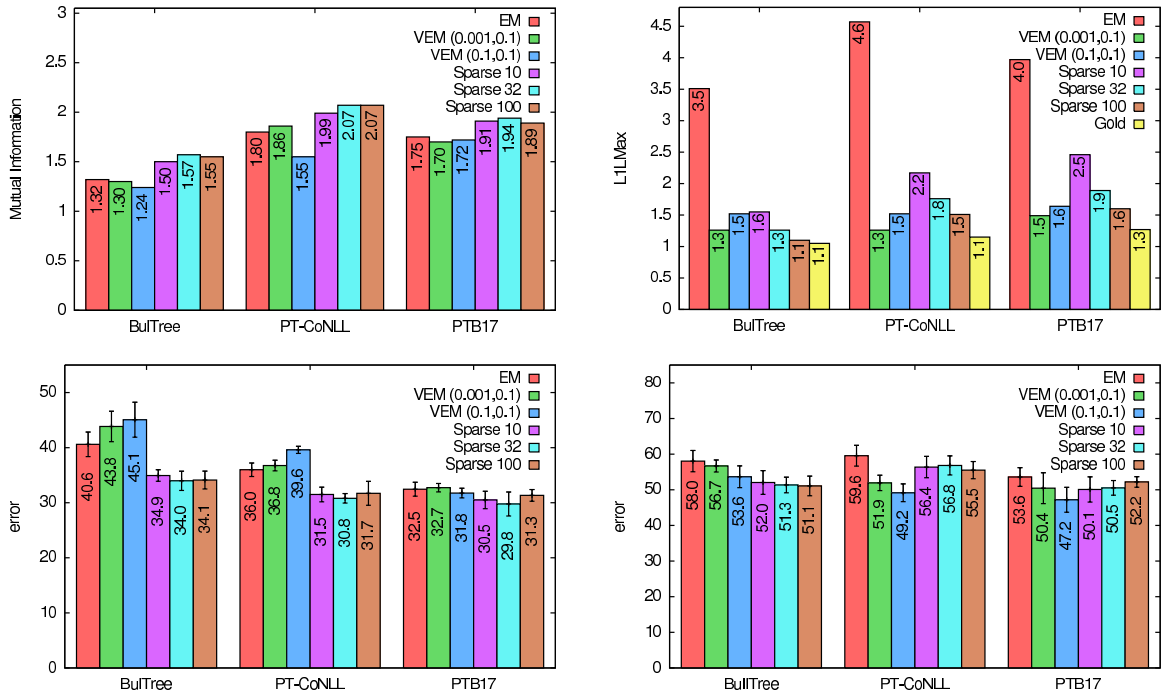


Figure 12: (Top Left) Mutual information in bits between gold tag distribution and hidden state distribution. The maximum is the entropy of the gold set (BulTree 3.05, PT-CoNLL 3.49 and PTB17 3.22), (Top Right) ℓ_1/ℓ_∞ value, and average (Bottom Left) 1-Many error, (Bottom Right) 1-1 error over 10 different runs (same seeds used for each model) for 200 iterations. Error bars are standard deviation for the 10 runs. All models are first order HMMs: EM trained using expectation maximization, VEM trained using variational EM using 0.1 state to state prior and (01,0.0001) observation prior; Sparse trained using PR with constraint strength $\sigma = 10, 32, 100$.

difference is bigger for VEM when the observation prior is set to bigger values (0.1), leading at the same time to worse results in 1-Many mapping.

9. Conclusion

In this paper we have presented posterior regularization (PR), a technique for regularizing models by encoding prior knowledge in constraints on model posteriors. On the algorithmic side, we have shown that PR can be solved efficiently in dual form, and that the regularization can be easily incorporated into a variant of the classical EM optimization method. In relating PR to similar frameworks, we have clarified its main advantages: faster optimization speed with respect to generalized expectation (Mann and McCallum, 2007, 2008), and greater distributional estimation accuracy with respect to constraint-driven learning (Chang et al., 2007). To the best of our knowledge, we are the first to link all these learning frameworks by explicitly stating a sense in which they all approximate the Bayesian perspective that motivates Liang et al. (2009). An interesting avenue for future work

includes an exploration of the tradeoff between computational complexity and accuracy of the different approximations presented in this and related work (Figure 4). For example, is there a large performance drop as we go from GE to PR and from PR to CODL, or are the variational and MAP approximations accurate in practice?

In addition to discussing PR's theoretical potential, we have demonstrated that it lives up to this potential in a wide variety realistic applications. The applications we focus on in this paper are word alignment, multi-view learning, dependency parsing, and part-of-speech tagging. Yet PR can express such a wide variety of prior knowledge that can be encoded by functions of model posteriors, and there remains a vast array of unexplored possible applications for this technique.

In addition to using PR in other applications, we would like to investigate alternative optimization methods. The main optimization bottleneck that PR implementations encounter is the extensive time required for projecting the posterior distribution into the constrained posterior space. Each evaluation of the objective or its gradient requires inference in the original model. One direction for exploration is the use of second order or approximate second-order optimization methods. Another potential direction is to use approximate inference in some parts of the optimization, for example fully factored variational inference. Finally, for applications where some constraints span multiple instances, but others do not not, it would be interesting to combine online and batch methods.

A last key extension to the current PR work is to explore the case where the constraint set Q is not easily specified using linear constraints on some constraint features ϕ. Thus far we have only developed theory and applications for linear constraints. It would be interesting to explore applications and derive efficient learning methods when the constraints are not linear, for example, applications with semi-definite or polynomial constraints.

Acknowledgments

The authors would like to thank Gregory Druck, Gideon Mann, Percy Liang, Fernando Pereira, Umar Syed and Mitch Marcus for helpful comments and discussion on drafts of this paper. J. V. Graça was supported by a fellowship from Fundação para a Ciência e Tecnologia (SFRH/ BD/ 27528/ 2006) and by FCT project CMU-PT/HuMach/0039/2008. K. Ganchev was supported by ARO MURI SUBTLE W911NF-07-1-0216. J. Gillenwater work was partially supported by NSF-IGERT 0504487. B. Taskar was partially supported by DARPA CSSG and ONR Young Investigator Award N000141010746.

Appendix A. Scaling the Strength of PR

This appendix describes how to optimize a version of our objective with scaled posterior regularization strength. In this case, we will use a modified EM algorithm that maximizes:

$$F'(q,\theta) = \mathcal{L}(\theta) - \alpha \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y} \mid \mathbf{X})) \qquad \text{s.t.}\, q \in Q$$

where $\alpha \in [0,1]$. The optimization procedure closely follows the one in Section 2.6. When performing the M-step, we use a mixture of the projected posteriors q and the model posteriors $p_\theta(\mathbf{Y}|\mathbf{X})$ to

update the model parameters. The updated EM algorithm is:

$$\mathbf{E}' - \mathbf{step}: \max_q F'(q,\theta) = \min_{q \in Q} \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})),$$
$$\mathbf{M}' - \mathbf{step}: \max_\theta F'(q,\theta) = \max_\theta (1-\alpha)\mathbf{E}_{p_{\theta'}}[\log p_\theta(\mathbf{X},\mathbf{Y})] + \alpha \mathbf{E}_q[\log p_\theta(\mathbf{X},\mathbf{Y})].$$

Note that the $\mathbf{E}'$-step is identical to the one in Equation 8.

Appendix B. Proof of Proposition 2.1

The modified E-step involves a projection step that minimizes the Kullback-Leibler divergence:

$$\arg\min_{q,\xi} \quad \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \qquad \text{s.t.} \quad \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi; \quad ||\xi||_\beta \leq \varepsilon.$$

Assuming the set $Q = \{q(\mathbf{Y}) : \exists \xi : \mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} \leq \xi; \quad ||\xi||_\beta \leq \varepsilon\}$ is non-empty, the corresponding Lagrangian is

$$\max_{\lambda \geq 0, \alpha \geq 0, \gamma} \min_{q(\mathbf{Y}), \xi} L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma),$$

where

$$L(q,\xi,\lambda,\alpha,\gamma) = \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \quad + \lambda \cdot (\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} - \xi) \\ + \alpha(||\xi||_\beta - \varepsilon) \quad + \gamma \left(\sum_{\mathbf{Y}} q(\mathbf{Y}) - 1 \right).$$

In order to compute the dual of this Lagrangian, we first represent

$$\alpha ||\xi||_\beta = \max_\eta \xi \cdot \eta \quad \text{s.t.} \quad ||\eta||_{\beta^*} \leq \alpha.$$

This results in a variational Lagrangian

$$\max_{\lambda \geq 0, \alpha \geq 0, \gamma} \max_{||\eta||_{\beta^*} \leq \alpha} \min_{q(\mathbf{Y}), \xi} L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta),$$

with $L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta)$ defined as

$$L(q,\xi,\lambda,\alpha,\gamma,\eta) = \quad \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \quad + \lambda \cdot (\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{b} - \xi) \\ + \xi \cdot \eta - \alpha\varepsilon \quad + \gamma \left(\sum_{\mathbf{Y}} q(\mathbf{Y}) - 1 \right),$$

$$\frac{\partial L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta)}{\partial q(\mathbf{Y})} = \quad \log q(\mathbf{Y}) + 1 - \log p_\theta(\mathbf{Y}|\mathbf{X}) + \lambda \cdot \phi(\mathbf{X},\mathbf{Y}) + \gamma = 0$$
$$\implies \quad q(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\lambda \cdot \phi(\mathbf{X},\mathbf{Y}))}{e \exp(\gamma)},$$

$$\frac{\partial L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta)}{\partial \xi_i} = \quad \eta_i - \lambda_i = 0 \quad \implies \quad \eta = \lambda. \qquad (26)$$

Note that Equation 26 implies that we have the constraint $||\lambda||_{\beta^*} \leq \alpha$ and also the positive and negative $\lambda \cdot \xi$ cancel each other out. Plugging $q(\mathbf{Y})$, $\eta = \lambda$ in $L(q(\mathbf{Y}), \xi, \lambda, \alpha, \gamma, \eta)$ and taking the derivative with respect to γ

$$\frac{\partial L(\lambda, \alpha, \gamma)}{\partial \gamma} = \sum_{\mathbf{Y}} \frac{p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\lambda \cdot \phi(\mathbf{X}, \mathbf{Y}))}{e \exp(\gamma)} - 1 = 0$$
$$\implies \gamma = \log\left(\frac{\sum_{\mathbf{Y}} p_\theta(\mathbf{Y}|\mathbf{X}) \exp\left(-\lambda \cdot \phi(\mathbf{x}, \mathbf{z})\right)}{e}\right).$$

From there we can simplify $q(\mathbf{Y}) = \frac{p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\lambda \cdot \phi(\mathbf{X}, \mathbf{Y}))}{Z_\lambda}$ where $Z_\lambda = \sum_{\mathbf{Y}} p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\lambda \cdot \phi(\mathbf{X}, \mathbf{Y}))$ ensures that $q(\mathbf{Y})$ is properly normalized. Plugging γ into $L(\lambda, \alpha, \gamma)$

$$L(\lambda, \alpha) = -\log(Z_\lambda) - \mathbf{b} \cdot \lambda - \alpha\varepsilon.$$

Now our objective is:

$$\max_{\lambda \geq 0, \alpha \geq 0} \quad -\log(Z_\lambda) - \mathbf{b} \cdot \lambda - \alpha\varepsilon \quad \text{s.t.} \quad ||\lambda||_{\beta^*} \leq \alpha.$$

We can analytically see that the optimum of this objective with respect to α is $\alpha = ||\lambda||_{\beta^*}$ and placing this in $L(\lambda, \alpha)$ we get the dual objective:

$$\textbf{Dual } \mathbf{E}' : \quad \arg\max_{\lambda \geq 0} \quad -\mathbf{b} \cdot \lambda - \log(Z_\lambda) - \varepsilon ||\lambda||_{\beta^*}$$

as desired.

References

A. Abeillé. *Treebanks: Building and Using Parsed Corpora*. Springer, 2003.

S. Afonso, E. Bick, R. Haber, and D. Santos. Floresta Sinta(c)tica: a treebank for Portuguese. In *Proc. LREC*, 2002.

Y. Altun, M. Johnson, and T. Hofmann. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proc. EMNLP*, 2003.

R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

J. Atserias, B. Casas, E. Comelles, M. González, L. Padró, and M. Padró. Freeling 1.3: Syntactic and semantic services in an open-source nlp library. In *Proc. LREC*, 2006.

M. Balcan and A. Blum. A PAC-style model for learning from labeled and unlabeled data. In *Proc. COLT*, 2005.

C. Bannard and C. Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proc. ACL*, 2005.

K. Bellare, G. Druck, and A. McCallum. Alternating projections for learning with expectation constraints. In *Proc. UAI*, 2009.

D. P. Bertsekas. *Nonlinear Programming: 2nd Edition*. Athena scientific, 1999.

J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proc. EMNLP*, 2006.

J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. ACL*, 2007.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. COLT*, 1998.

U. Brefeld, C. Büscher, and T. Scheffer. Multi-view hidden markov perceptrons. In *Proc. LWA*, 2005.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, M. J. Goldsmith, J. Hajic, R. L. Mercer, and S. Mohanty. But dictionaries are data too. In *Proc. HLT*, 1993.

P. F. Brown, S. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1994.

C. Callison-Burch. *Paraphrasing and Translation*. PhD thesis, University of Edinburgh, 2007.

C. Callison-Burch. Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. In *Proc. EMNLP*, 2008.

A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr., and T. M. Mitchell. Coupled Semi-Supervised Learning for Information Extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM)*, 2010.

M. Chang, L. Ratinov, and D. Roth. Guiding semi-supervision with constraint-driven learning. In *Proc. ACL*, 2007.

M.W. Chang, L. Ratinov, N. Rizzolo, and D. Roth. Learning and inference with constraints. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. AAAI, 2008.

C. Chelba, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. Ristad, R. Rosenfeld, A. Stolcke, and D. Wu. Structure and performance of a dependency language model. In *Proc. Eurospeech*, 1997.

D. Chiang, A. Lopez, N. Madnani, C. Monz, P. Resnik, and M. Subotin. The hiero machine translation system: extensions, evaluation, and analysis. In *Proc. HLT-EMNLP*, 2005.

M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proc. SIGDAT-EMNLP*, 1999.

H. Daumé III. Cross-task knowledge-constrained self training. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2008.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Royal Statistical Society, Ser. B*, 39(1):1–38, 1977.

G. Druck, G. Mann, and A. McCallum. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *Proc. ACL-IJCNLP*, 2009.

J. Eisner. Three new probabilistic models for dependency parsing: an exploration. In *Proc. CoLing*, 1996.

H. Fox. Phrasal cohesion and statistical machine translation. In *Proc. EMNLP*, 2002.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. What's in a translation rule? In *Proc. HLT-NAACL*, 2004.

K. Ganchev, J. Graça, J. Blitzer, and B. Taskar. Multi-view learning over structured and non-identical outputs. In *Proc. UAI*, 2008a.

K. Ganchev, J. Graça, and B. Taskar. Better alignments = better translations? In *Proc. ACL*, 2008b.

K. Ganchev, J. Gillenwater, and B. Taskar. Dependency grammar induction via bitext projection constraints. In *Proc. ACL-IJCNLP*, 2009.

J. Gao and M. Johnson. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proc. EMNLP*, 2008.

S. Goldwater and T. Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proc. ACL*, 2007.

J. Graça, K. Ganchev, and B. Taskar. Expectation maximization and posterior constraints. In *Proc. NIPS*, 2007.

J. Graça, K. Ganchev, F. Pereira, and B. Taskar. Parameter vs. posterior sparisty in latent variable models. In *Proc. NIPS*, 2009a.

J. Graça, K. Ganchev, and B. Taskar. Postcat - posterior constrained alignment toolkit. In *The Third Machine Translation Marathon*, 2009b.

J. Graça, K. Ganchev, and B. Taskar. Learning tractable word alignment models with complex constraints. *Computational Linguistics*, 36, September 2010.

A. Haghighi and D. Klein. Prototype-driven learning for sequence models. In *Proc. NAACL*, 2006.

A. Haghighi, A. Ng, and C. Manning. Robust textual inference via graph matching. In *Proc. EMNLP*, 2005.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:11–311, 2005.

M. Johnson. Why doesn't EM find good HMM POS-taggers. In *Proc. EMNLP-CoNLL*, 2007.

T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communications*, 15(1):52–60, 2 1967. ISSN 0096-2244.

S. Kakade and D. Foster. Multi-view regression via canonical correlation analysis. In *Proc. COLT*, 2007.

D. Klein and C. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. ACL*, 2004.

P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, 2005.

P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proc. NAACL*, 2003.

S. Lee and K. Choi. Reestimation and best-first parsing algorithm for probabilistic dependency grammar. In *Proc. WVLC-5*, 1997.

Z. Li and J. Eisner. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 40–51, Singapore, 2009.

P. Liang, B. Taskar, and D. Klein. Alignment by agreement. In *Proc. HLT-NAACL*, 2006.

P. Liang, M. I. Jordan, and D. Klein. Learning from measurements in exponential families. In *Proc. ICML*, 2009.

G. S. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proc. ICML*, 2007.

G. S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proc. ACL*, 2008.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

E. Matusov, R. Zens, and H. Ney. Symmetric word alignments for statistical machine translation. In *Proc. COLING*, 2004.

E. Matusov, N. Ueffing, and H. Ney. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. EACL*, 2006.

R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proc. ACL*, 2005.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer, 1998.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proc. EMNLP-CoNLL*, 2007.

F. J. Och and H. Ney. Improved statistical alignment models. In *Proc. ACL*, 2000.

F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003. ISSN 0891-2017.

A. Pauls, J. Denero, and D. Klein. Consensus training for consensus decoding in machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1418–1427, Singapore, 2009. Association for Computational Linguistics.

N. Quadrianto, J. Petterson, and A. Smola. Distribution matching for transduction. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1500–1508. MIT Press, 2009.

C. Quirk, A. Menezes, and C. Cherry. Dependency treelet translation: syntactically informed phrasal smt. In *Proc. ACL*, 2005.

M. Rogati, S. McCarley, and Y. Yang. Unsupervised learning of arabic stemming using a parallel corpus. In *Proc. ACL*, 2003.

D. Rosenberg and P. Bartlett. The rademacher complexity of co-regularized kernel classes. In *Proc. AI Stats*, 2007.

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. CoNLL and LLL*, 2000.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proc. HLT-NAACL*, 2003.

L. Shen, J. Xu, and R. Weischedel. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. ACL*, 2008.

K. Simov, P. Osenova, M. Slavcheva, S. Kolkovska, E. Balabanova, D. Doikoff, K. Ivanova, A. Simov, E. Simov, and M. Kouylekov. Building a linguistically interpreted corpus of bulgarian: the bultreebank. In *Proc. LREC*, 2002.

V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Proc. ICML*, 2005.

A. Smith, T. Cohn, and M. Osborne. Logarithmic opinion pools for conditional random fields. In *Proc. ACL*, 2005.

B. Snyder and R. Barzilay. Unsupervised multilingual learning for morphological segmentation. In *Proc. ACL*, 2008.

B. Snyder, T. Naseem, J. Eisenstein, and R. Barzilay. Adding more languages improves unsupervised multilingual part-of-speech tagging: a bayesian non-parametric approach. In *Proc. NAACL*, 2009.

J. Tiedemann. Building a multilingual parallel subtitle corpus. In *Proc. CLIN*, 2007.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. HLT-NAACL*, 2003.

P. Tseng. An analysis of the EM algorithm and entropy-like proximal point methods. *Mathematics of Operations Research*, 29(1):27–44, 2004.

Y. Tsuruoka and J. Tsujii. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proc. HLT-EMNLP*, 2005.

L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8: 189–201, 1979.

S. Vogel, H. Ney, and C. Tillmann. Hmm-based word alignment in statistical translation. In *Proc. COLING*, 1996.

H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proc. IWPT*, 2003.

D. Yarowsky and G. Ngai. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proc. NAACL*, 2001.

Journal of Machine Learning Research 11 (2010) 2051-2055 Submitted 3/10; Revised 6/10; Published 7/10

A Surrogate Modeling and Adaptive Sampling Toolbox for Computer Based Design

Dirk Gorissen DIRK.GORISSEN@UGENT.BE
Ivo Couckuyt IVO.COUCKUYT@UGENT.BE
Piet Demeester PIET.DEMEESTER@UGENT.BE
Tom Dhaene TOM.DHAENE@UGENT.BE
Ghent University - IBBT
Department of Information Technology (INTEC)
Gaston Crommenlaan 8 bus 201
9050 Gent, Belgium

Karel Crombecq KAREL.CROMBECQ@UA.AC.BE
University of Antwerp
Department of Maths and Computer Science
Middelheimlaan 1
2020 Antwerpen, Belgium

Editor: Cheng Soon Ong

Abstract

An exceedingly large number of scientific and engineering fields are confronted with the need for computer simulations to study complex, real world phenomena or solve challenging design problems. However, due to the computational cost of these high fidelity simulations, the use of neural networks, kernel methods, and other surrogate modeling techniques have become indispensable. Surrogate models are compact and cheap to evaluate, and have proven very useful for tasks such as optimization, design space exploration, prototyping, and sensitivity analysis. Consequently, in many fields there is great interest in tools and techniques that facilitate the construction of such regression models, while minimizing the computational cost and maximizing model accuracy. This paper presents a mature, flexible, and adaptive machine learning toolkit for regression modeling and active learning to tackle these issues. The toolkit brings together algorithms for data fitting, model selection, sample selection (active learning), hyperparameter optimization, and distributed computing in order to empower a domain expert to efficiently generate an accurate model for the problem or data at hand.

Keywords: surrogate modeling, metamodeling, function approximation, model selection, adaptive sampling, active learning, distributed computing

1. Background and Motivation

In many science and engineering problems researchers make heavy use of computer simulation codes in order to replace expensive physical experiments and improve the quality and performance of engineered products and devices. Such simulation activities are collectively referred to as computational science/engineering. Unfortunately, while allowing scientists more flexibility to study phenomena under controlled conditions, computer simulations require a substantial investment of

computation time. One simulation may take many minutes, hours, days or even weeks, quickly rendering parameter studies impractical (Forrester et al., 2008; Simpson et al., 2008).

Of the different ways to deal with this problem, this paper is concerned with the construction of simpler approximation models to predict the system performance and develop a relationship between the system inputs and outputs. When properly constructed, these approximation models mimic the behavior of the simulation accurately while being computationally cheap(er) to evaluate. Different approximation methods exist, each with their relative merits. This work concentrates on the use of data-driven, global approximations using compact surrogate models (also known as metamodels, replacement models, or response surface models). Examples include: rational functions, Kriging models, Artificial Neural Networks (ANN), splines, and Support Vector Machines (SVM). Once such a global approximation is available it is of great use for gaining insight into the behavior of the underlying system. The surrogate may be easily queried, optimized, visualized, and seamlessly integrated into CAD/CAE software packages.

The challenge is thus how to generate an approximation model that is as accurate as possible over the *complete* domain of interest while minimizing the simulation cost. Solving this challenge involves multiple sub-problems that must be addressed: how to interface with the simulation code, how to run simulations (locally, or on a cluster or cloud), which model type to approximate the data with and how to set the model complexity (e.g., topology of a neural network), how to estimate the model quality and ensure the domain expert trusts the model, how to decide which simulations to run (data collection), etc. The data collection aspect is worth emphasizing. Since data is computationally expensive to obtain and the optimal data distribution is not known up front, data points should be selected iteratively, there where the information gain will be the greatest. A sampling function is needed that minimizes the number of sample points selected in each iteration, yet maximizes the information gain of each iteration step. This process is called adaptive sampling but is also known as active learning, or sequential design.

There is a complex dependency web between these different options and dealing with these dependencies is non-trivial, particularly for a domain expert for whom the surrogate model is just an intermediate step towards solving a larger, more important problem. Few domain experts will be experts in the intricacies of efficient sampling and modeling strategies. Their primary concern is obtaining an accurate replacement metamodel for their problem as fast as possible and with minimal overhead (Gorissen et al., 2009d). As a result these choices are often made in a pragmatic, sometimes even ad-hoc, manner.

This paper discusses an advanced, and integrated software framework that provides a flexible and rigorous means to tackle such problems. This work lies at the intersection of Machine Learning/AI, Modeling and Simulation, and Distributed Computing. The methods developed are applicable to any domain where a cheap, accurate, approximation is needed to replace some expensive reference model. Our experience has been that the availability of such a framework can facilitate the transfer of knowledge from surrogate modeling researchers and lower the barrier of entry for domain experts.

2. SUMO Toolbox

The platform in question is the Matlab SUrrogate MOdeling (SUMO) Toolbox, illustrated in Figure 1. Given a simulation engine (Fluent, Cadence, Abaqus, HFSS, etc.) or other data source (data

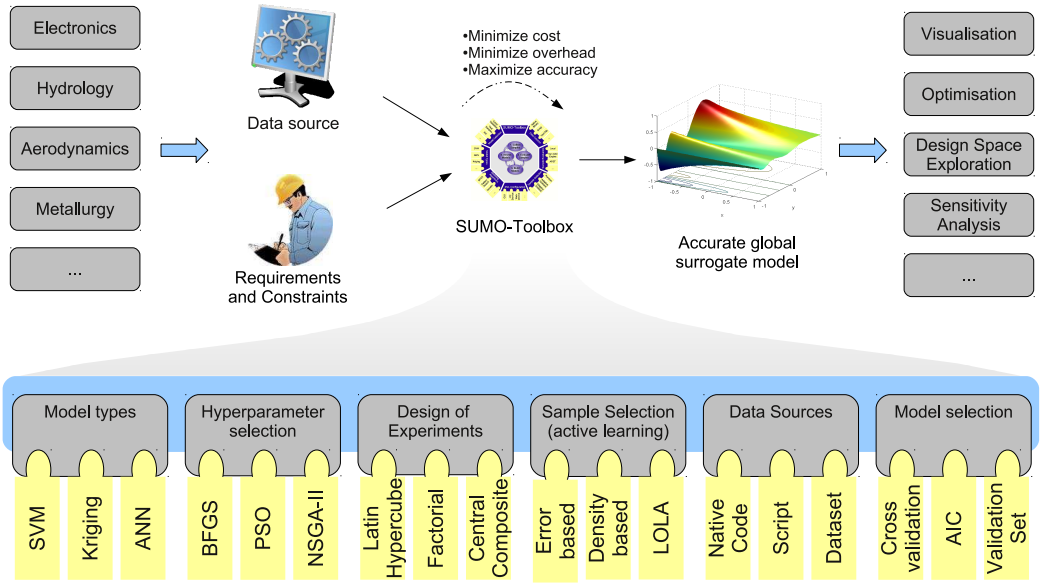


Figure 1: The SUMO Toolbox is a flexible framework for accurate global surrogate modeling and adaptive sampling (active learning). It features a rich set of plugins, is applicable to a wide range of domains, and can be applied in an autonomous, black-box fashion, or under full manual control. Written in Matlab and Java it is fully cross platform and comes with a large (60+) number of example problems.

set, Matlab script, Java class, etc.), the toolbox drives the data source to produce a surrogate model within the time and accuracy constraints set by the user.

The SUMO Toolbox adopts a microkernel design philosophy with many different plugins available for each of the different sub-problems:[1] model types (rational functions, Kriging, splines, SVM, ANN, etc.), hyperparameter optimization algorithms (Particle Swarm Optimization, Efficient Global Optimization, simulated annealing, Genetic Algorithm, etc.), model selection algorithms (cross validation, AIC, Leave-out set, etc.), sample selection (random, error based, density based, hybrid, etc.), Design of Experiments (Latin hypercube, Box-Bhenken, etc.), and sample evaluation methods (local, on a cluster or grid). The behavior of each software component is configurable through a central XML file and components can easily be added, removed or replaced by custom implementations. In addition the toolbox provides 'meta' plugins. For example to automatically select the best model type for a given problem (Gorissen et al., 2009d) or to use multiple model selection or sample selection criteria in concert (Gorissen et al., 2010).

Furthermore, there is built-in support for high performance computing. On the modeling side, the model generation process can take full advantage of multi-core CPUs and even of a complete cluster or grid. This can result in significant speedups for model types where the fitting process can be expensive (e.g., neural networks). Likewise, sample evaluation (simulation) can occur locally (with the option to take advantage of multi-core architectures) or on a separate compute cluster or grid (possibly accessed through a remote head-node). All interfacing with the grid middleware

1. The full list of plugins and features can be found at `http://www.sumowiki.intec.ugent.be`.

(submission, job monitoring, rescheduling of failed/lost simulation points, etc.) is handled transparently and automatically (see Gorissen et al., 2009c for more details). Also, the sample evaluation component runs in parallel with the other components (non-blocking) and not sequentially. This allows for an optimal use of computational resources.

In addition the SUMO Toolbox contains extensive logging and profiling capabilities so that the modeling process can easily be tracked and the modeling decisions understood. Once a final model has been generated, a GUI tool is available to visually explore the model (including derivatives and prediction uncertainty), assess its quality, and export it for use in other software tools.

3. Applications

The SUMO Toolbox has already been applied successfully to a very wide range of applications, including RF circuit block modeling (Gorissen et al., 2009b), hydrological modeling (Couckuyt et al., 2009), Electronic Packaging (Zhu and Franzon, 2009), aerodynamic modeling (Gorissen et al., 2009a), process engineering (Stephens et al., 2009), and automotive data modeling (Gorissen et al., 2010). Besides global modeling capabilities, the SUMO Toolbox also includes a powerful optimization framework based on the Efficient Global Optimization framework developed by Jones et al. (1998). As of version 6.1, the toolbox also contains an example of how the framework can also be applied to solve classification problems.

In sum, the goal of the toolbox is to fill the void in machine learning software when it comes to the challenging, costly, real-valued, problems faced in computational engineering. The toolbox is in use successfully at various institutions and we are continuously refining and extending the set of available plugins as the number of applications increase. Usage instructions, design documentation, and stable releases for all major platforms can be found at `http://www.sumo.intec.ugent.be`.

References

I. Couckuyt, D. Gorissen, H. Rouhani, E. Laermans, and T. Dhaene. Evolutionary regression modeling with active learning: An application to rainfall runoff modeling. In *International Conference on Adaptive and Natural Computing Algorithms*, volume LNCS 5495, pages 548–558, Sep. 2009.

A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, 2008.

D. Gorissen, K. Crombecq, I. Couckuyt, and T. Dhaene. *Foundations of Computational Intelligence, Volume 1: Learning and Approximation: Theoretical Foundations and Applications*, volume 201, chapter Automatic approximation of expensive functions with active learning, pages 35–62. Springer Verlag, Series Studies in Computational Intelligence, 2009a.

D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene. Sequential modeling of a low noise amplifier with neural networks and active learning. *Neural Computing and Applications*, 18(5): 485–494, Jun. 2009b.

D. Gorissen, T. Dhaene, P. Demeester, and J. Broeckhove. *Handbook of Research on Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Applications*, chapter Grid enabled surrogate modeling, pages 249–258. IGI Global, May 2009c.

D. Gorissen, T. Dhaene, and F. DeTurck. Evolutionary model type selection for global surrogate modeling. *Journal of Machine Learning Research*, 10:2039–2078, 2009d.

D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene. Multiobjective global surrogate modeling,dealing with the 5-percent problem. *Engineering with Computers*, 26(1):81–89, Jan. 2010.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, Nov. 1998. ISSN 0925-5001.

T. W. Simpson, V. Toropov, V. Balabanov, and F. A. C. Viana. Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come or not. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008 MAO, Victoria, Canada*, 2008.

D.W. Stephens, D. Gorissen, and T. Dhaene. Surrogate based sensitivity analysis of process equipment. In *Proc. of 7th International Conference on CFD in the Minerals and Process Industries, CSIRO, Melbourne, Australia*, Dec. 2009.

T. Zhu and P. D. Franzon. Application of surrogate modeling to generate compact and PVT-sensitive IBIS models. In *Proceedings of the 18th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct. 2009.

Journal of Machine Learning Research 11 (2010) 2057-2078 Submitted 6/09; Revised 4/10; Published 7/10

Matrix Completion from Noisy Entries

Raghunandan H. Keshavan RAGHURAM@STANFORD.EDU
Andrea Montanari* MONTANARI@STANFORD.EDU
Sewoong Oh SWOH@STANFORD.EDU
Department of Electrical Engineering
Stanford University
Stanford, CA 94304, USA

Editor: Tommi Jaakkola

Abstract

Given a matrix M of low-rank, we consider the problem of reconstructing it from noisy observations of a small, random subset of its entries. The problem arises in a variety of applications, from collaborative filtering (the 'Netflix problem') to structure-from-motion and positioning. We study a low complexity algorithm introduced by Keshavan, Montanari, and Oh (2010), based on a combination of spectral techniques and manifold optimization, that we call here OPTSPACE. We prove performance guarantees that are order-optimal in a number of circumstances.

Keywords: matrix completion, low-rank matrices, spectral methods, manifold optimization

1. Introduction

Spectral techniques are an authentic workhorse in machine learning, statistics, numerical analysis, and signal processing. Given a matrix M, its largest singular values—and the associated singular vectors—'explain' the most significant correlations in the underlying data source. A low-rank approximation of M can further be used for low-complexity implementations of a number of linear algebra algorithms (Frieze et al., 2004).

In many practical circumstances we have access only to a sparse subset of the entries of an $m \times n$ matrix M. It has recently been discovered that, if the matrix M has rank r, and unless it is too 'structured', a small random subset of its entries allow to reconstruct it *exactly*. This result was first proved by Candès and Recht (2008) by analyzing a convex relaxation introduced by Fazel (2002). A tighter analysis of the same convex relaxation was carried out by Candès and Tao (2009). A number of iterative schemes to solve the convex optimization problem appeared soon thereafter (Cai et al., 2008; Ma et al., 2009; Toh and Yun, 2009).

In an alternative line of work, Keshavan, Montanari, and Oh (2010) attacked the same problem using a combination of spectral techniques and manifold optimization: We will refer to their algorithm as OPTSPACE. OPTSPACE is intrinsically of low complexity, the most complex operation being computing r singular values (and the corresponding singular vectors) of a sparse $m \times n$ matrix. The performance guarantees proved by Keshavan et al. (2010) are comparable with the information theoretic lower bound: roughly $nr \max\{r, \log n\}$ random entries are needed to reconstruct M exactly (here we assume m of order n). A related approach was also developed by Lee and Bresler (2009), although without performance guarantees for matrix completion.

*. Also in Department of Statistics.

The above results crucially rely on the assumption that M is *exactly* a rank r matrix. For many applications of interest, this assumption is unrealistic and it is therefore important to investigate their robustness. Can the above approaches be generalized when the underlying data is 'well approximated' by a rank r matrix? This question was addressed by Candès and Plan (2009) within the convex relaxation approach of Candès and Recht (2008). The present paper proves a similar robustness result for OPTSPACE. Remarkably the guarantees we obtain are order-optimal in a variety of circumstances, and improve over the analogous results of Candès and Plan (2009).

1.1 Model Definition

Let M be an $m \times n$ matrix of rank r, that is

$$M = U\Sigma V^T . \tag{1}$$

where U has dimensions $m \times r$, V has dimensions $n \times r$, and Σ is a diagonal $r \times r$ matrix. We assume that each entry of M is perturbed, thus producing an 'approximately' low-rank matrix N, with

$$N_{ij} = M_{ij} + Z_{ij} ,$$

where the matrix Z will be assumed to be 'small' in an appropriate sense.

Out of the $m \times n$ entries of N, a subset $E \subseteq [m] \times [n]$ is revealed. We let N^E be the $m \times n$ matrix that contains the revealed entries of N, and is filled with 0's in the other positions

$$N^E_{ij} = \begin{cases} N_{ij} & \text{if } (i,j) \in E , \\ 0 & \text{otherwise.} \end{cases}$$

Analogously, we let M^E and Z^E be the $m \times n$ matrices that contain the entries of M and Z, respectively, in the revealed positions and is filled with 0's in the other positions. The set E will be uniformly random given its size $|E|$.

1.2 Algorithm

For the reader's convenience, we recall the algorithm introduced by Keshavan et al. (2010), which we will analyze here. The basic idea is to minimize the cost function $F(X,Y)$, defined by

$$\begin{aligned} F(X,Y) &\equiv \min_{S \in \mathbb{R}^{r\times r}} \mathcal{F}(X,Y,S) , \\ \mathcal{F}(X,Y,S) &\equiv \frac{1}{2} \sum_{(i,j)\in E} (N_{ij} - (XSY^T)_{ij})^2 . \end{aligned} \tag{2}$$

Here $X \in \mathbb{R}^{n\times r}$, $Y \in \mathbb{R}^{m\times r}$ are orthogonal matrices, normalized by $X^TX = m\mathbf{I}$, $Y^TY = n\mathbf{I}$.

Minimizing $F(X,Y)$ is an *a priori* difficult task, since F is a non-convex function. The key insight is that the singular value decomposition (SVD) of N^E provides an excellent initial guess, and that the minimum can be found with high probability by standard gradient descent after this initialization. Two caveats must be added to this description: (1) In general the matrix N^E must be 'trimmed' to eliminate over-represented rows and columns; (2) For technical reasons, we consider a slightly modified cost function to be denoted by $\widetilde{F}(X,Y)$.

OPTSPACE(matrix N^E)

1: Trim N^E, and let $\widetilde{N}^E$ be the output;
2: Compute the rank-r projection of $\widetilde{N}^E$, $\mathsf{P}_r(\widetilde{N}^E) = X_0 S_0 Y_0^T$;
3: Minimize $\widetilde{F}(X,Y)$ through gradient descent, with initial condition (X_0, Y_0).

We may note here that the rank of the matrix M, if not known, can be reliably estimated from $\widetilde{N}^E$ (Keshavan and Oh, 2009).

The various steps of the above algorithm are defined as follows.

Trimming. We say that a row is 'over-represented' if it contains more than $2|E|/m$ revealed entries (i.e., more than twice the average number of revealed entries per row). Analogously, a column is over-represented if it contains more than $2|E|/n$ revealed entries. The trimmed matrix $\widetilde{N}^E$ is obtained from N^E by setting to 0 over-represented rows and columns.

Rank-r projection. Let

$$\widetilde{N}^E = \sum_{i=1}^{\min(m,n)} \sigma_i x_i y_i^T ,$$

be the singular value decomposition of $\widetilde{N}^E$, with singular values $\sigma_1 \geq \sigma_2 \geq \dots$. We then define

$$\mathsf{P}_r(\widetilde{N}^E) = \frac{mn}{|E|} \sum_{i=1}^{r} \sigma_i x_i y_i^T .$$

Apart from an overall normalization, $\mathsf{P}_r(\widetilde{N}^E)$ is the best rank-r approximation to $\widetilde{N}^E$ in Frobenius norm.

Minimization. The modified cost function $\widetilde{F}$ is defined as

$$\begin{aligned} \widetilde{F}(X,Y) &= F(X,Y) + \rho\, G(X,Y) \\ &\equiv F(X,Y) + \rho \sum_{i=1}^{m} G_1\left(\frac{\|X^{(i)}\|^2}{3\mu_0 r}\right) + \rho \sum_{j=1}^{n} G_1\left(\frac{\|Y^{(j)}\|^2}{3\mu_0 r}\right) , \end{aligned}$$

where $X^{(i)}$ denotes the i-th row of X, and $Y^{(j)}$ the j-th row of Y. The function $G_1 : \mathbb{R}^+ \to \mathbb{R}$ is such that $G_1(z) = 0$ if $z \leq 1$ and $G_1(z) = e^{(z-1)^2} - 1$ otherwise. Further, we can choose $\rho = \Theta(|E|)$.

Let us stress that the regularization term is mainly introduced for our proof technique to work (and a broad family of functions G_1 would work as well). In numerical experiments we did not find any performance loss in setting $\rho = 0$.

One important feature of OPTSPACE is that $F(X,Y)$ and $\widetilde{F}(X,Y)$ are regarded as functions of the r-dimensional subspaces of $\mathbb{R}^m$ and $\mathbb{R}^n$ generated (respectively) by the columns of X and Y. This interpretation is justified by the fact that $F(X,Y) = F(XA, YB)$ for any two orthogonal matrices $A, B \in \mathbb{R}^{r\times r}$ (the same property holds for $\widetilde{F}$). The set of r dimensional subspaces of $\mathbb{R}^m$ is a differentiable Riemannian manifold $\mathsf{G}(m,r)$ (the Grassmann manifold). The gradient descent algorithm is applied to the function $\widetilde{F} : \mathsf{M}(m,n) \equiv \mathsf{G}(m,r) \times \mathsf{G}(n,r) \to \mathbb{R}$. For further details on optimization by gradient descent on matrix manifolds we refer to Edelman et al. (1999) and Absil et al. (2008).

1.3 Some Notations

The matrix M to be reconstructed takes the form (1) where $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$. We write $U = [u_1, u_2, \dots, u_r]$ and $V = [v_1, v_2, \dots, v_r]$ for the columns of the two factors, with $\|u_i\| = \sqrt{m}$, $\|v_i\| = \sqrt{n}$, and $u_i^T u_j = 0$, $v_i^T v_j = 0$ for $i \neq j$ (there is no loss of generality in this, since normalizations can be absorbed by redefining Σ).

We shall write $\Sigma = \mathrm{diag}(\Sigma_1, \dots, \Sigma_r)$ with $\Sigma_1 \geq \Sigma_2 \geq \cdots \geq \Sigma_r > 0$. The maximum and minimum singular values will also be denoted by $\Sigma_{\max} = \Sigma_1$ and $\Sigma_{\min} = \Sigma_r$. Further, the maximum size of an entry of M is $M_{\max} \equiv \max_{ij} |M_{ij}|$.

Probability is taken with respect to the uniformly random subset $E \subseteq [m] \times [n]$ given $|E|$ and (eventually) the noise matrix Z. Define $\varepsilon \equiv |E|/\sqrt{mn}$. In the case when $m = n$, ε corresponds to the average number of revealed entries per row or column. Then it is convenient to work with a model in which each entry is revealed independently with probability $\varepsilon/\sqrt{mn}$. Since, with high probability $|E| \in [\varepsilon\sqrt{\alpha}n - A\sqrt{n\log n}, \varepsilon\sqrt{\alpha}n + A\sqrt{n\log n}]$, any guarantee on the algorithm performances that holds within one model, holds within the other model as well if we allow for a vanishing shift in ε. We will use C, C' etc. to denote universal numerical constants.

It is convenient to define the following projection operator $\mathcal{P}_E(\cdot)$ as the sampling operator, which maps an $m \times n$ matrix onto an $|E|$-dimensional subspace in $\mathbb{R}^{m\times n}$

$$\mathcal{P}_E(N)_{ij} = \begin{cases} N_{ij} & \text{if } (i,j) \in E\,, \\ 0 & \text{otherwise.} \end{cases}$$

Given a vector $x \in \mathbb{R}^n$, $\|x\|$ will denote its Euclidean norm. For a matrix $X \in \mathbb{R}^{n\times n'}$, $\|X\|_F$ is its Frobenius norm, and $\|X\|_2$ its operator norm (i.e., $\|X\|_2 = \sup_{u\neq 0} \|Xu\|/\|u\|$). The standard scalar product between vectors or matrices will sometimes be indicated by $\langle x, y\rangle$ or $\langle X, Y\rangle \equiv \mathrm{Tr}(X^T Y)$, respectively. Finally, we use the standard combinatorics notation $[n] = \{1, 2, \dots, n\}$ to denote the set of first n integers.

1.4 Main Results

Our main result is a performance guarantee for OPTSPACE under appropriate incoherence assumptions, and is presented in Section 1.4.2. Before presenting it, we state a theorem of independent interest that provides an error bound on the simple trimming-plus-SVD approach. The reader interested in the OPTSPACE guarantee can go directly to Section 1.4.2.

Throughout this paper, without loss of generality, we assume $\alpha \equiv m/n \geq 1$.

1.4.1 SIMPLE SVD

Our first result shows that, in great generality, the rank-r projection of $\widetilde{N}^E$ provides a reasonable approximation of M. We define $\widetilde{Z}^E$ to be an $m \times n$ matrix obtained from Z^E, after the trimming step of the pseudocode above, that is, by setting to zero the over-represented rows and columns.

Theorem 1.1 *Let $N = M + Z$, where M has rank r, and assume that the subset of revealed entries $E \subseteq [m] \times [n]$ is uniformly random with size $|E|$. Let $M_{\max} = \max_{(i,j)\in[m]\times[n]} |M_{ij}|$. Then there exists numerical constants C and C' such that*

$$\frac{1}{\sqrt{mn}} \|M - \mathsf{P}_r(\widetilde{N}^E)\|_F \leq C M_{\max} \left(\frac{nr\alpha^{3/2}}{|E|}\right)^{1/2} + C' \frac{n\sqrt{r\alpha}}{|E|} \|\widetilde{Z}^E\|_2\,,$$

with probability larger than $1-1/n^3$.

Projection onto rank-r matrices through SVD is a pretty standard tool, and is used as first analysis method for many practical problems. At a high-level, projection onto rank-r matrices can be interpreted as 'treat missing entries as zeros'. This theorem shows that this approach is reasonably robust if the number of observed entries is as large as the number of degrees of freedom (which is about $(m+n)r$) times a large constant. The error bound is the sum of two contributions: the first one can be interpreted as an undersampling effect (error induced by missing entries) and the second as a noise effect. Let us stress that trimming is crucial for achieving this guarantee.

1.4.2 OPTSPACE

Theorem 1.1 helps to set the stage for the key point of this paper: *a much better approximation is obtained by minimizing the cost* $\widetilde{F}(X,Y)$ *(step 3 in the pseudocode above), provided* M *satisfies an appropriate incoherence condition*. Let $M = U\Sigma V^T$ be a low rank matrix, and assume, without loss of generality, $U^TU = m\mathbf{I}$ and $V^TV = n\mathbf{I}$. We say that M is (μ_0,μ_1)-*incoherent* if the following conditions hold.

A1. For all $i \in [m]$, $j \in [n]$ we have, $\sum_{k=1}^r U_{ik}^2 \le \mu_0 r$, $\sum_{k=1}^r V_{ik}^2 \le \mu_0 r$.

A2. For all $i \in [m]$, $j \in [n]$ we have, $|\sum_{k=1}^r U_{ik}(\Sigma_k/\Sigma_1)V_{jk}| \le \mu_1 r^{1/2}$.

Theorem 1.2 *Let* $N = M + Z$, *where* M *is a* (μ_0,μ_1)-*incoherent matrix of rank* r, *and assume that the subset of revealed entries* $E \subseteq [m]\times[n]$ *is uniformly random with size* $|E|$. *Further, let* $\Sigma_{\min} = \Sigma_r \le \cdots \le \Sigma_1 = \Sigma_{\max}$ *with* $\Sigma_{\max}/\Sigma_{\min} \equiv \kappa$. *Let* $\widehat{M}$ *be the output of* OPTSPACE *on input* N^E. *Then there exists numerical constants* C *and* C' *such that if*

$$|E| \;\ge\; Cn\sqrt{\alpha}\kappa^2 \max\left\{\mu_0 r\sqrt{\alpha}\log n\,;\,\mu_0^2 r^2\alpha\kappa^4\,;\,\mu_1^2 r^2\alpha\kappa^4\right\}\,,$$

then, with probability at least $1-1/n^3$,

$$\frac{1}{\sqrt{mn}}\|\widehat{M}-M\|_F \le C'\kappa^2\frac{n\sqrt{r\alpha}}{|E|}\|Z^E\|_2\,. \tag{3}$$

provided that the right-hand side is smaller than $\Sigma_{\min}$.

As discussed in the next section, this theorem captures rather sharply the effect of important classes of noise on the performance of OPTSPACE.

1.5 Noise Models

In order to make sense of the above results, it is convenient to consider a couple of simple models for the noise matrix Z:

Independent entries model. We assume that Z's entries are i.i.d. random variables, with zero mean $\mathbb{E}\{Z_{ij}\} = 0$ and sub-Gaussian tails. The latter means that

$$\mathbb{P}\{|Z_{ij}| \ge x\} \le 2\,e^{-\frac{x^2}{2\sigma^2}}\,,$$

for some constant σ^2 uniformly bounded in n.

Worst case model. In this model Z is arbitrary, but we have an uniform bound on the size of its entries: $|Z_{ij}| \le Z_{\max}$.

The basic parameter entering our main results is the operator norm of $\widetilde{Z}^E$, which is bounded as follows in these two noise models.

Theorem 1.3 *If Z is a random matrix drawn according to the independent entries model, then for any sample size $|E|$ there is a constant C such that,*

$$\|\widetilde{Z}^E\|_2 \le C\sigma \left(\frac{|E|\log n}{n}\right)^{1/2}, \tag{4}$$

with probability at least $1-1/n^3$. Further there exists a constant C' such that, if the sample size is $|E| \ge n\log n$ (for $n \ge \alpha$), we have

$$\|\widetilde{Z}^E\|_2 \le C'\sigma \left(\frac{|E|}{n}\right)^{1/2}, \tag{5}$$

with probability at least $1-1/n^3$.

If Z is a matrix from the worst case model, then

$$\|\widetilde{Z}^E\|_2 \le \frac{2|E|}{n\sqrt{\alpha}} Z_{\max},$$

for any realization of E.

It is elementary to show that, if $|E| \ge 15\alpha n\log n$, no row or column is over-represented with high probability. It follows that in the regime of $|E|$ for which the conditions of Theorem 1.2 are satisfied, we have $Z^E = \widetilde{Z}^E$ and hence the bound (5) applies to $\|\widetilde{Z}^E\|_2$ as well. Then, among the other things, this result implies that for the independent entries model the right-hand side of our error estimate, Eq. (3), is with high probability smaller than $\Sigma_{\min}$, if $|E| \ge Cr\alpha n\kappa^4(\sigma/\Sigma_{\min})^2$. For the worst case model, the same statement is true if $Z_{\max} \le \Sigma_{\min}/C\sqrt{r}\kappa^2$.

1.6 Comparison with Other Approaches to Matrix Completion

Let us begin by mentioning that a statement analogous to our preliminary Theorem 1.1 was proved by Achlioptas and McSherry (2007). Our result however applies to any number of revealed entries, while the one of Achlioptas and McSherry (2007) requires $|E| \ge (8\log n)^4 n$ (which for $n \le 5\cdot 10^8$ is larger than n^2). We refer to Section 1.8 for further discussion of this point.

As for Theorem 1.2, we will mainly compare our algorithm with the convex relaxation approach recently analyzed by Candès and Plan (2009), and based on semidefinite programming. Our basic setting is indeed the same, while the algorithms are rather different.

Figures 1 and 2 compare the average root mean square error $\|\widehat{M}-M\|_F/\sqrt{mn}$ for the two algorithms as a function of $|E|$ and the rank-r respectively. Here M is a random rank r matrix of dimension $m = n = 600$, generated by letting $M = \widetilde{U}\widetilde{V}^T$ with $\widetilde{U}_{ij}, \widetilde{V}_{ij}$ i.i.d. $N(0, 20/\sqrt{n})$. The noise is distributed according to the independent noise model with $Z_{ij} \sim N(0,1)$. In the first suite of simulations, presented in Figure 1, the rank is fixed to $r = 2$. In the second one (Figure 2), the number of samples is fixed to $|E| = 72000$. These examples are taken from Candès and Plan (2009, Figure

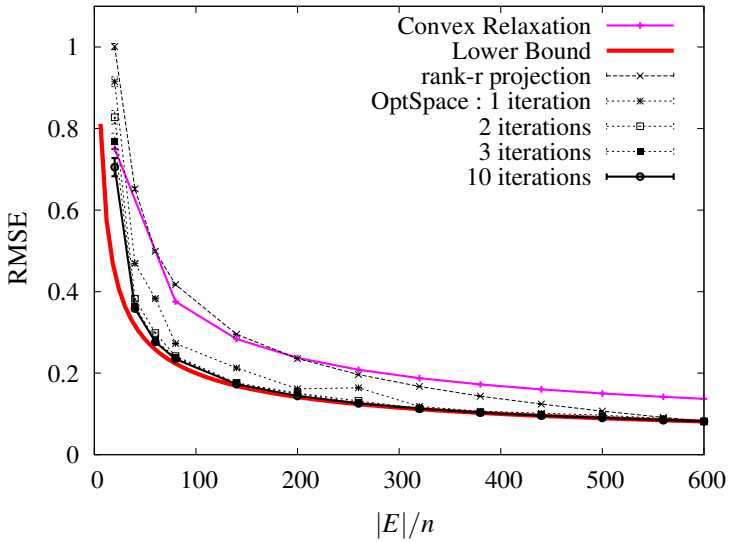


Figure 1: Numerical simulation with random rank-2 600×600 matrices. Root mean square error achieved by OPTSPACE is shown as a function of the number of observed entries $|E|$ and of the number of line minimizations. The performance of nuclear norm minimization and an information theoretic lower bound are also shown.

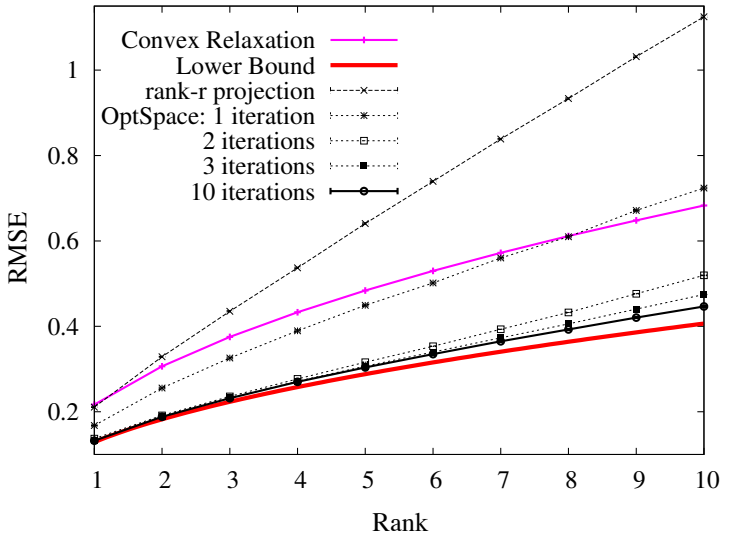


Figure 2: Numerical simulation with random rank-r 600×600 matrices and number of observed entries $|E|/n = 120$. Root mean square error achieved by OPTSPACE is shown as a function of the rank and of the number of line minimizations. The performance of nuclear norm minimization and an information theoretic lower bound are also shown.

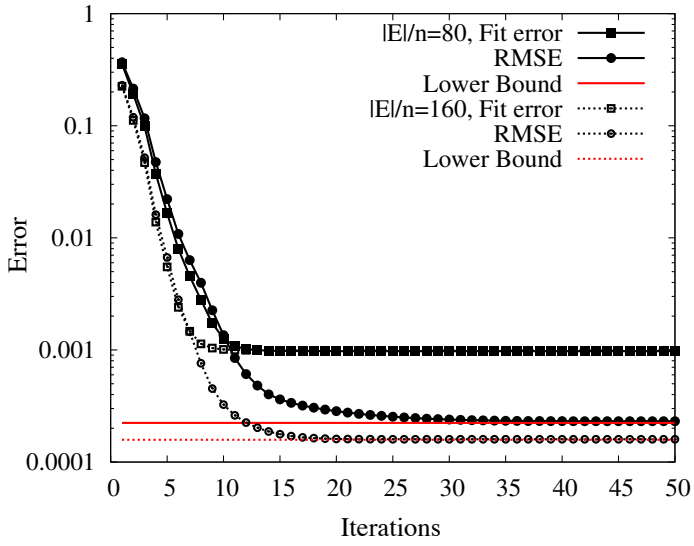


Figure 3: Numerical simulation with random rank-2 600×600 matrices and number of observed entries $|E|/n = 80$ and 160. The standard deviation of the i.i.d. Gaussian noise is 0.001. Fit error and root mean square error achieved by OPTSPACE are shown as functions of the number of line minimizations. Information theoretic lower bounds are also shown.

2), from which we took the data points for the convex relaxation approach, as well as the information theoretic lower bound described later in this section. After a few iterations, OPTSPACE has a smaller root mean square error than the one produced by convex relaxation. In about 10 iterations it becomes indistinguishable from the information theoretic lower bound for small ranks.

In Figure 3, we illustrate the rate of convergence of OPTSPACE. Two metrics, root mean squared error(RMSE) and fit error $\|\mathcal{P}_E(\widehat{M}-N)\|_F/\sqrt{|E|}$, are shown as functions of the number of iterations in the manifold optimization step. Note, that the fit error can be easily evaluated since $N^E = \mathcal{P}_E(N)$ is always available at the estimator. M is a random 600×600 rank-2 matrix generated as in the previous examples. The additive noise is distributed as $Z_{ij} \sim N(0, \sigma^2)$ with $\sigma = 0.001$ (A small noise level was used in order to trace the RMSE evolution over many iterations). Each point in the figure is the averaged over 20 random instances, and resulting errors for two different values of sample size $|E| = 80$ and $|E| = 160$ are shown. In both cases, we can see that the RMSE converges to the information theoretic lower bound described later in this section. The fit error decays exponentially with the number iterations and converges to the standard deviation of the noise which is 0.001. This is a lower bound on the fit error when $r \ll n$, since even if we have a perfect reconstruction of M, the average fit error is still 0.001.

For a more complete numerical comparison between various algorithms for matrix completion, including different noise models, real data sets and ill conditioned matrices, we refer to Keshavan and Oh (2009).

Next, let us compare our main result with the performance guarantee of Candès and Plan (2009, Theorem 7). Let us stress that we require the condition number κ to be bounded, while the analysis of Candès and Plan (2009) and Candès and Tao (2009) requires a stronger incoherence assumption

(compared to our **A1**). Therefore the assumptions are not directly comparable. As far as the error bound is concerned, Candès and Plan (2009) proved that the semidefinite programming approach returns an estimate $\widehat{M}$ which satisfies

$$\frac{1}{\sqrt{mn}}\|\widehat{M}_{\mathrm{SDP}}-M\|_F \le 7\sqrt{\frac{n}{|E|}}\|Z^E\|_F+\frac{2}{n\sqrt{\alpha}}\|Z^E\|_F\,. \tag{6}$$

(The constant in front of the first term is in fact slightly smaller than 7 in Candès and Plan (2009), but in any case larger than $4\sqrt{2}$. We choose to quote a result which is slightly less accurate but easier to parse.)

Theorem 1.2 improves over this result in several respects: (1) We do not have the second term on the right-hand side of (6), that actually increases with the number of observed entries; (2) Our error decreases as $n/|E|$ rather than $(n/|E|)^{1/2}$; (3) The noise enters Theorem 1.2 through the operator norm $\|Z^E\|_2$ instead of its Frobenius norm $\|Z^E\|_F \ge \|Z^E\|_2$. For E uniformly random, one expects $\|Z^E\|_F$ to be roughly of order $\|Z^E\|_2\sqrt{n}$. For instance, within the independent entries model with bounded variance σ, $\|Z^E\|_F=\Theta(\sqrt{|E|})$ while $\|Z^E\|_2$ is of order $\sqrt{|E|/n}$ (up to logarithmic terms).

Theorem 1.2 can also be compared to an information theoretic lower bound computed by Candès and Plan (2009). Suppose, for simplicity, $m=n$ and assume that an oracle provides us a linear subspace T where the correct rank r matrix $M=U\Sigma V^T$ lies. More precisely, we know that $M\in T$ where T is a linear space of dimension $2nr-r^2$ defined by

$$T=\{UY^T+XV^T \mid X\in\mathbb{R}^{n\times r}, Y\in\mathbb{R}^{n\times r}\}\,.$$

Notice that the rank constraint is therefore replaced by this simple linear constraint. The minimum mean square error estimator is computed by projecting the revealed entries onto the subspace T, which can be done by solving a least squares problem. Candès and Plan (2009) analyzed the root mean squared error of the resulting estimator $\widehat{M}$ and showed that

$$\frac{1}{\sqrt{mn}}\|\widehat{M}_{\mathrm{Oracle}}-M\|_F \approx \sqrt{\frac{1}{|E|}}\|Z^E\|_F\,.$$

Here $\approx$ indicates that the root mean squared error concentrates in probability around the right-hand side.

For the sake of comparison, suppose we have i.i.d. Gaussian noise with variance σ^2. In this case the oracle estimator yields (for $r=o(n)$)

$$\frac{1}{\sqrt{mn}}\|\widehat{M}_{\mathrm{Oracle}}-M\|_F \approx \sigma\sqrt{\frac{2nr}{|E|}}\,.$$

The bound (6) on the semidefinite programming approach yields

$$\frac{1}{\sqrt{mn}}\|\widehat{M}_{\mathrm{SDP}}-M\|_F \le \sigma\left(7\sqrt{n|E|}+\frac{2}{n}|E|\right).$$

Finally, using Theorems 1.2 and 1.3 we deduce that OPTSPACE achieves

$$\frac{1}{\sqrt{mn}}\|\widehat{M}_{\mathrm{OptSpace}}-M\|_F \le \sigma\sqrt{\frac{Cnr}{|E|}}\,.$$

Hence, when the noise is i.i.d. Gaussian with small enough σ, OPTSPACE is order-optimal.

1.7 Related Work on Gradient Descent

Local optimization techniques such as gradient descent of coordinate descent have been intensively studied in machine learning, with a number of applications. Here we will briefly review the recent literature on the use of such techniques within collaborative filtering applications.

Collaborative filtering was studied from a graphical models perspective in Salakhutdinov et al. (2007), which introduced an approach to prediction based on Restricted Boltzmann Machines (RBM). Exact learning of the model parameters is intractable for such models, but the authors studied the performances of a *contrastive divergence*, which computes an approximate gradient of the likelihood function, and uses it to optimize the likelihood locally. Based on empirical evidence, it was argued that RBM's have several advantages over spectral methods for collaborative filtering.

An objective function analogous to the one used in the present paper was considered early on in Srebro and Jaakkola (2003), which uses gradient descent in the factors to minimize a weighted sum of square residuals. Salakhutdinov and Mnih (2008) justified the use of such an objective function by deriving it as the (negative) log-posterior of an appropriate probabilistic model. This approach naturally lead to the use of quadratic regularization in the factors. Again, gradient descent in the factors was used to perform the optimization. Also, this paper introduced a logistic mapping between the low-rank matrix and the recorded ratings.

Recently, this line of work was pushed further in Salakhutdinov and Srebro (2010), which emphasize the advantage of using a non-uniform quadratic regularization in the factors. The basic objective function was again a sum of square residuals, and version of stochastic gradient descent was used to optimize it.

This rich and successful line of work emphasizes the importance of obtaining a rigorous understanding of methods based on local minimization of the sum of square residuals with respect to the factors. The present paper provides a first step in that direction. Hopefully the techniques developed here will be useful to analyze the many variants of this approach.

The relationship between the non-convex objective function and convex relaxation introduced by Fazel (2002) was further investigated by Srebro et al. (2005) and Recht et al. (2007). The basic relation is provided by the identity

$$\|M\|_* = \frac{1}{2} \min_{M=XY^T} \left\{ \|X\|_F^2 + \|Y\|_F^2 \right\}, \tag{7}$$

where $\|M\|_*$ denotes the nuclear norm of M (the sum of its singular values). In other words, adding a regularization term that is quadratic in the factors (as the one used in much of the literature reviewed above) is equivalent to weighting M by its nuclear norm, that can be regarded as a convex surrogate of its rank.

In view of the identity (7) it might be possible to use the results in this paper to prove stronger guarantees on the nuclear norm minimization approach. Unfortunately this implication is not immediate. Indeed in the present paper we assume the correct rank r is known, while on the other hand we do not use a quadratic regularization in the factors. (See Keshavan and Oh, 2009 for a procedure that estimates the rank from the data and is provably successful under the hypotheses of Theorem 1.2.) Trying to establish such an implication, and clarifying the relation between the two approaches is nevertheless a promising research direction.

1.8 On the Spectrum of Sparse Matrices and the Role of Trimming

The trimming step of the OPTSPACE algorithm is somewhat counter-intuitive in that we seem to be wasting information. In this section we want to clarify its role through a simple example. Before describing the example, let us stress once again two facts: (i) In the last step of our the algorithm, the trimmed entries are actually incorporated in the cost function and hence the full information is exploited; (ii) Trimming is not the only way to treat over-represented rows/columns in M^E, and probably not the optimal one. One might for instance rescale the entries of such rows/columns. We stick to trimming because we can prove it actually works.

Let us now turn to the example. Assume, for the sake of simplicity, that $m = n$, there is no noise in the revealed entries, and M is the rank one matrix with $M_{ij} = 1$ for all i and j. Within the independent sampling model, the matrix M^E has i.i.d. entries, with distribution Bernoulli(ε/n). The number of non-zero entries in a column is Binomial$(n, \varepsilon/n)$ and is independent for different columns. It is not hard to realize that the column with the largest number of entries has more than $C \log n/\log\log n$ entries, with positive probability (this probability can be made as large as we want by reducing C). Let i be the index of this column, and consider the test vector $\underline{e}^{(i)}$ that has the i-th entry equal to 1 and all the others equal to 0. By computing $\|M^E \underline{e}^{(i)}\|$, we conclude that the largest singular value of M^E is at least $\sqrt{C \log n/\log\log n}$. In particular, this is very different from the largest singular value of $\mathbb{E}\{M^E\} = (\varepsilon/n)M$ which is ε. This suggests that approximating M with the $\mathsf{P}_r(M^E)$ leads to a large error. Hence trimming is crucial in proving Theorem 1.1. Also, the phenomenon is more severe in real data sets than in the present model, where each entry is revealed independently.

Trimming is also crucial in proving Theorem 1.3. Using the above argument, it is possible to show that under the worst case model,

$$\|Z^E\|_2 \geq C'(\varepsilon)\, Z_{\max} \sqrt{\frac{\log n}{\log\log n}}\,.$$

This suggests that the largest singular value of the noise matrix Z^E is quite different from the largest singular value of $\mathbb{E}\{Z^E\}$ which is $\varepsilon Z_{\max}$.

To summarize, Theorems 1.1 and 1.3 (for the worst case model) simply do not hold without trimming or a similar procedure to normalize rows/columns of N^E. Trimming allows to overcome the above phenomenon by setting to 0 over-represented rows/columns.

2. Proof of Theorem 1.1

As explained in the introduction, the crucial idea is to consider the singular value decomposition of the trimmed matrix $\widetilde{N}^E$ instead of the original matrix N^E. Apart from a trivial rescaling, these singular values are close to the ones of the original matrix M.

Lemma 1 *There exists a numerical constant C such that, with probability greater than $1 - 1/n^3$,*

$$\left|\frac{\sigma_q}{\varepsilon} - \Sigma_q\right| \leq C M_{\max}\sqrt{\frac{\alpha}{\varepsilon}} + \frac{1}{\varepsilon}\|\widetilde{Z}^E\|_2\,,$$

where it is understood that $\Sigma_q = 0$ for $q > r$.

Proof For any matrix A, let $\sigma_q(A)$ denote the qth singular value of A. Then, $\sigma_q(A+B) \le \sigma_q(A) + \sigma_1(B)$, whence

$$\begin{aligned}\left|\frac{\sigma_q}{\varepsilon} - \Sigma_q\right| &\le \left|\frac{\sigma_q(\widetilde{M}^E)}{\varepsilon} - \Sigma_q\right| + \frac{\sigma_1(\widetilde{Z}^E)}{\varepsilon} \\ &\le CM_{\max}\sqrt{\frac{\alpha}{\varepsilon}} + \frac{1}{\varepsilon}\|\widetilde{Z}^E\|_2\,,\end{aligned}$$

where the second inequality follows from the next Lemma as shown by Keshavan et al. (2010).

Lemma 2 (Keshavan, Montanari, Oh, 2009) *There exists a numerical constant C such that, with probability larger than* $1 - 1/n^3$,

$$\frac{1}{\sqrt{mn}}\left|\left|M - \frac{\sqrt{mn}}{\varepsilon}\widetilde{M}^E\right|\right|_2 \le CM_{\max}\sqrt{\frac{\alpha}{\varepsilon}}\,.$$

■

We will now prove Theorem 1.1.

Proof (Theorem 1.1) For any matrix A of rank at most $2r$, $\|A\|_F \le \sqrt{2r}\|A\|_2$, whence

$$\begin{aligned}\frac{1}{\sqrt{mn}}\|M - \mathsf{P}_r(\widetilde{N}^E)\|_F &\le \frac{\sqrt{2r}}{\sqrt{mn}}\left|\left|M - \frac{\sqrt{mn}}{\varepsilon}\left(\widetilde{N}^E - \sum_{i\ge r+1}\sigma_i x_i y_i^T\right)\right|\right|_2 \\ &= \frac{\sqrt{2r}}{\sqrt{mn}}\left|\left|M - \frac{\sqrt{mn}}{\varepsilon}\left(\widetilde{M}^E + \widetilde{Z}^E - \sum_{i\ge r+1}\sigma_i x_i y_i^T\right)\right|\right|_2 \\ &= \frac{\sqrt{2r}}{\sqrt{mn}}\left|\left|\left(M - \frac{\sqrt{mn}}{\varepsilon}\widetilde{M}^E\right) + \frac{\sqrt{mn}}{\varepsilon}\left(\widetilde{Z}^E - \Big(\sum_{i\ge r+1}\sigma_i x_i y_i^T\Big)\right)\right|\right|_2 \\ &\le \frac{\sqrt{2r}}{\sqrt{mn}}\left(\left|\left|M - \frac{\sqrt{mn}}{\varepsilon}\widetilde{M}^E\right|\right|_2 + \frac{\sqrt{mn}}{\varepsilon}\|\widetilde{Z}^E\|_2 + \frac{\sqrt{mn}}{\varepsilon}\sigma_{r+1}\right) \\ &\le 2CM_{\max}\sqrt{\frac{2\alpha r}{\varepsilon}} + \frac{2\sqrt{2r}}{\varepsilon}\|\widetilde{Z}^E\|_2 \\ &\le C'M_{\max}\left(\frac{nr\alpha^{3/2}}{|E|}\right)^{1/2} + 2\sqrt{2}\left(\frac{n\sqrt{r\alpha}}{|E|}\right)\|\widetilde{Z}^E\|_2\,.\end{aligned}$$

where on the fourth line, we have used the fact that for any matrices A_i, $\|\sum_i A_i\|_2 \le \sum_i \|A_i\|_2$. This proves our claim. ■

3. Proof of Theorem 1.2

Recall that the cost function is defined over the Riemannian manifold $\mathsf{M}(m,n) \equiv \mathsf{G}(m,r) \times \mathsf{G}(n,r)$. The proof of Theorem 1.2 consists in controlling the behavior of F in a neighborhood of $\mathbf{u} = (U,V)$ (the point corresponding to the matrix M to be reconstructed). Throughout the proof we let $\mathcal{K}(\mu)$ be the set of matrix couples $(X,Y) \in \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$ such that $\|X^{(i)}\|^2 \le \mu r$, $\|Y^{(j)}\|^2 \le \mu r$ for all i, j.

3.1 Preliminary Remarks and Definitions

Given $\mathbf{x}_1 = (X_1, Y_1)$ and $\mathbf{x}_2 = (X_2, Y_2) \in \mathsf{M}(m,n)$, two points on this manifold, their distance is defined as $d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{d(X_1, X_2)^2 + d(Y_1, Y_2)^2}$, where, letting $(\cos\theta_1, \dots, \cos\theta_r)$ be the singular values of $X_1^T X_2/m$,

$$d(X_1, X_2) = \|\theta\|_2 .$$

The next remark bounds the distance between two points on the manifold. In particular, we will use this to bound the distance between the original matrix $M = U\Sigma V^T$ and the starting point of the manifold optimization $\widehat{M} = X_0 S_0 Y_0^T$.

Remark 3 (Keshavan, Montanari, Oh, 2009) *Let $U, X \in \mathbb{R}^{m\times r}$ with $U^T U = X^T X = m\mathbf{I}$, $V, Y \in \mathbb{R}^{n\times r}$ with $V^T V = Y^T Y = n\mathbf{I}$, and $M = U\Sigma V^T$, $\widehat{M} = XSY^T$ for $\Sigma = \mathrm{diag}(\Sigma_1, \dots, \Sigma_r)$ and $S \in \mathbb{R}^{r\times r}$. If $\Sigma_1, \dots, \Sigma_r \geq \Sigma_{\min}$, then*

$$d(U, X) \leq \frac{\pi}{\sqrt{2\alpha} n \Sigma_{\min}} \|M - \widehat{M}\|_F \quad , \quad d(V, Y) \leq \frac{\pi}{\sqrt{2\alpha} n \Sigma_{\min}} \|M - \widehat{M}\|_F$$

Given S achieving the minimum in Eq. (2), it is also convenient to introduce the notations

$$d_-(\mathbf{x}, \mathbf{u}) \equiv \sqrt{\Sigma_{\min}^2 d(\mathbf{x}, \mathbf{u})^2 + \|S - \Sigma\|_F^2} ,$$
$$d_+(\mathbf{x}, \mathbf{u}) \equiv \sqrt{\Sigma_{\max}^2 d(\mathbf{x}, \mathbf{u})^2 + \|S - \Sigma\|_F^2} .$$

3.2 Auxiliary Lemmas and Proof of Theorem 1.2

The proof is based on the following two lemmas that generalize and sharpen analogous bounds in Keshavan et al. (2010).

Lemma 4 *There exist numerical constants C_0, C_1, C_2 such that the following happens. Assume $\varepsilon \geq C_0 \mu_0 r \sqrt{\alpha} \max\{\log n ; \mu_0 r \sqrt{\alpha} (\Sigma_{\min}/\Sigma_{\max})^4\}$ and $\delta \leq \Sigma_{\min}/(C_0 \Sigma_{\max})$. Then,*

$$F(\mathbf{x}) - F(\mathbf{u}) \geq C_1 n \varepsilon \sqrt{\alpha}\, d_-(\mathbf{x}, \mathbf{u})^2 - C_1 n \sqrt{r\alpha} \|Z^E\|_2 d_+(\mathbf{x}, \mathbf{u}) , \quad (8)$$
$$F(\mathbf{x}) - F(\mathbf{u}) \leq C_2 n \varepsilon \sqrt{\alpha} \Sigma_{\max}^2 d(\mathbf{x}, \mathbf{u})^2 + C_2 n \sqrt{r\alpha} \|Z^E\|_2 d_+(\mathbf{x}, \mathbf{u}) , \quad (9)$$

for all $\mathbf{x} \in \mathsf{M}(m,n) \cap \mathcal{K}(4\mu_0)$ such that $d(\mathbf{x}, \mathbf{u}) \leq \delta$, with probability at least $1 - 1/n^4$. Here $S \in \mathbb{R}^{r\times r}$ is the matrix realizing the minimum in Eq. (2).

Corollary 3.1 *There exist a constant C such that, under the hypotheses of Lemma 4*

$$\|S - \Sigma\|_F \leq C \Sigma_{\max} d(\mathbf{x}, \mathbf{u}) + C \frac{\sqrt{r}}{\varepsilon} \|Z^E\|_2 .$$

Further, for an appropriate choice of the constants in Lemma 4, we have

$$\sigma_{\max}(S) \leq 2\Sigma_{\max} + C \frac{\sqrt{r}}{\varepsilon} \|Z^E\|_2 , \quad (10)$$
$$\sigma_{\min}(S) \geq \frac{1}{2} \Sigma_{\min} - C \frac{\sqrt{r}}{\varepsilon} \|Z^E\|_2 . \quad (11)$$

Lemma 5 *There exist numerical constants C_0, C_1, C_2 such that the following happens. Assume $\varepsilon \geq C_0 \mu_0 r \sqrt{\alpha}\, (\Sigma_{\max}/\Sigma_{\min})^2 \max\{\log n; \mu_0 r \sqrt{\alpha} (\Sigma_{\max}/\Sigma_{\min})^4\}$ and $\delta \leq \Sigma_{\min}/(C_0 \Sigma_{\max})$. Then,*

$$\|\mathrm{grad}\, \widetilde{F}(\mathbf{x})\|^2 \geq C_1\, n\varepsilon^2\, \Sigma_{\min}^4 \left[d(\mathbf{x},\mathbf{u}) - C_2 \frac{\sqrt{r}\Sigma_{\max}}{\varepsilon \Sigma_{\min}} \frac{\|Z^E\|_2}{\Sigma_{\min}} \right]_+^2 , \tag{12}$$

for all $\mathbf{x} \in \mathsf{M}(m,n) \cap \mathcal{K}(4\mu_0)$ such that $d(\mathbf{x},\mathbf{u}) \leq \delta$, with probability at least $1 - 1/n^4$. (Here $[a]_+ \equiv \max(a,0)$.)

We can now turn to the proof of our main theorem.

Proof (Theorem 1.2). Let $\delta = \Sigma_{\min}/C_0\Sigma_{\max}$ with C_0 large enough so that the hypotheses of Lemmas 4 and 5 are verified.

Call $\{\mathbf{x}_k\}_{k\geq 0}$ the sequence of pairs $(X_k, Y_k) \in \mathsf{M}(m,n)$ generated by gradient descent. By assumption the right-hand side of Eq. (3) is smaller than $\Sigma_{\min}$. The following is therefore true for some numerical constant C:

$$\|Z^E\|_2 \leq \frac{\varepsilon}{C\sqrt{r}} \left(\frac{\Sigma_{\min}}{\Sigma_{\max}} \right)^2 \Sigma_{\min} . \tag{13}$$

Notice that the constant appearing here can be made as large as we want by modifying the constant appearing in the statement of the theorem. Further, by using Corollary 3.1 in Eqs. (8) and (9) we get

$$F(\mathbf{x}) - F(\mathbf{u}) \geq C_1 n\varepsilon\sqrt{\alpha}\Sigma_{\min}^2 \left\{ d(\mathbf{x},\mathbf{u})^2 - \delta_{0,-}^2 \right\}, \tag{14}$$

$$F(\mathbf{x}) - F(\mathbf{u}) \leq C_2 n\varepsilon\sqrt{\alpha}\Sigma_{\max}^2 \left\{ d(\mathbf{x},\mathbf{u})^2 + \delta_{0,+}^2 \right\}, \tag{15}$$

with C_1 and C_2 different from those in Eqs. (8) and (9), where

$$\delta_{0,-} \equiv C \frac{\sqrt{r}\Sigma_{\max}}{\varepsilon \Sigma_{\min}} \frac{\|Z^E\|_2}{\Sigma_{\min}}, \qquad \delta_{0,+} \equiv C \frac{\sqrt{r}\Sigma_{\max}}{\varepsilon \Sigma_{\min}} \frac{\|Z^E\|_2}{\Sigma_{\max}} .$$

By Eq. (13), with large enough C, we can assume $\delta_{0,-} \leq \delta/20$ and $\delta_{0,+} \leq (\delta/20)(\Sigma_{\min}/\Sigma_{\max})$.

Next, we provide a bound on $d(\mathbf{u},\mathbf{x}_0)$. Using Remark 3, we have $d(\mathbf{u},\mathbf{x}_0) \leq (\pi/n\sqrt{\alpha}\Sigma_{\min})\|M - X_0 S_0 Y_0^T\|_F$. Together with Theorem 1.1 this implies

$$d(\mathbf{u},\mathbf{x}_0) \leq \frac{C M_{\max}}{\Sigma_{\min}} \left(\frac{r\alpha}{\varepsilon} \right)^{1/2} + \frac{C'\sqrt{r}}{\varepsilon \Sigma_{\min}} \|\widetilde{Z}^E\|_2 .$$

Since $\varepsilon \geq C'' \alpha \mu_1^2 r^2 (\Sigma_{\max}/\Sigma_{\min})^4$ as per our assumptions and $M_{\max} \leq \mu_1 \sqrt{r}\Sigma_{\max}$ for incoherent M, the first term in the above bound is upper bounded by $\Sigma_{\min}/20C_0\Sigma_{\max}$, for large enough C''. Using Eq. (13), with large enough constant C, the second term in the above bound is upper bounded by $\Sigma_{\min}/20C_0\Sigma_{\max}$. Hence we get

$$d(\mathbf{u},\mathbf{x}_0) \leq \frac{\delta}{10} .$$

We make the following claims :

1. $\mathbf{x}_k \in \mathcal{K}(4\mu_0)$ for all k.

   First we notice that we can assume $\mathbf{x}_0 \in \mathcal{K}(3\mu_0)$. Indeed, if this does not hold, we can 'rescale' those rows of X_0, Y_0 that violate the constraint. A proof that this rescaling is possible was given in Keshavan et al. (2010) (cf. Remark 6.2 there). We restate the result here for the reader's convenience in the next Remark.

   Remark 6 *Let $U, X \in \mathbb{R}^{n\times r}$ with $U^T U = X^T X = n\mathbf{I}$ and $U \in \mathcal{K}(\mu_0)$ and $d(X,U) \le \delta \le \frac{1}{16}$. Then there exists $X' \in \mathbb{R}^{n\times r}$ such that $X'^T X' = n\mathbf{I}$, $X' \in \mathcal{K}(3\mu_0)$ and $d(X',U) \le 4\delta$. Further, such an X' can be computed from X in a time of $O(nr^2)$.*

   Since $\mathbf{x}_0 \in \mathcal{K}(3\mu_0)$, $\widetilde{F}(\mathbf{x}_0) = F(\mathbf{x}_0) \le 4C_2 n\varepsilon\sqrt{\alpha}\Sigma_{\max}^2\delta^2/100$. On the other hand $\widetilde{F}(\mathbf{x}) \ge \rho(e^{1/9}-1)$ for $\mathbf{x} \notin \mathcal{K}(4\mu_0)$. Since $\widetilde{F}(\mathbf{x}_k)$ is a non-increasing sequence, the thesis follows provided we take $\rho \ge C_2 n\varepsilon\sqrt{\alpha}\Sigma_{\min}^2$.

2. $d(\mathbf{x}_k, \mathbf{u}) \le \delta/10$ for all k.

   Since $\varepsilon \ge C\alpha\mu_1^2 r^2(\Sigma_{\max}/\Sigma_{\min})^6$ as per our assumptions in Theorem 1.2, we have $d(\mathbf{x}_0,\mathbf{u})^2 \le (C_1\Sigma_{\min}^2/C_2\Sigma_{\max}^2)(\delta/20)^2$. Also assuming Eq. (13) with large enough C, we have $\delta_{0,-} \le \delta/20$ and $\delta_{0,+} \le (\delta/20)(\Sigma_{\min}/\Sigma_{\max})$. Then, by Eq. (15),

   $$F(\mathbf{x}_0) \le F(\mathbf{u}) + C_1 n\varepsilon\sqrt{\alpha}\Sigma_{\min}^2\frac{2\delta^2}{400}.$$

   Also, using Eq. (14), for all $\mathbf{x}_k$ such that $d(\mathbf{x}_k,\mathbf{u}) \in [\delta/10,\delta]$, we have

   $$F(\mathbf{x}) \ge F(\mathbf{u}) + C_1 n\varepsilon\sqrt{\alpha}\Sigma_{\min}^2\frac{3\delta^2}{400}.$$

   Hence, for all $\mathbf{x}_k$ such that $d(\mathbf{x}_k,\mathbf{u}) \in [\delta/10,\delta]$, we have $\widetilde{F}(\mathbf{x}) \ge F(\mathbf{x}) \ge F(\mathbf{x}_0)$. This contradicts the monotonicity of $\widetilde{F}(\mathbf{x})$, and thus proves the claim.

Since the cost function is twice differentiable, and because of the above two claims, the sequence $\{\mathbf{x}_k\}$ converges to

$$\Omega = \left\{\mathbf{x} \in \mathcal{K}(4\mu_0) \cap \mathsf{M}(m,n) : d(\mathbf{x},\mathbf{u}) \le \delta, \operatorname{grad}\widetilde{F}(\mathbf{x}) = 0\right\}.$$

By Lemma 5 for any $\mathbf{x} \in \Omega$,

$$d(\mathbf{x},\mathbf{u}) \le C\frac{\sqrt{r}\Sigma_{\max}}{\varepsilon\Sigma_{\min}}\frac{\|Z^E\|_2}{\Sigma_{\min}}. \tag{16}$$

Using Corollary 3.1, we have $d_+(\mathbf{x},\mathbf{u}) \le \Sigma_{\max} d(\mathbf{x},\mathbf{u}) + \|S-\Sigma\|_F \le C\Sigma_{\max} d(\mathbf{x},\mathbf{u}) + C(\sqrt{r}/\varepsilon)\|Z^E\|_2$. Together with Eqs. (18) and (16), this implies

$$\frac{1}{n\sqrt{\alpha}}\|M - XSY^T\|_F \le C\frac{\sqrt{r}\Sigma_{\max}^2\|Z^E\|_2}{\varepsilon\Sigma_{\min}^2},$$

which finishes the proof of Theorem 1.2. ■

3.3 Proof of Lemma 4 and Corollary 3.1

Proof (Lemma 4) The proof is based on the analogous bound in the noiseless case, that is, Lemma 5.3 in Keshavan et al. (2010). For readers' convenience, the result is reported in Appendix A, Lemma 7. For the proof of these lemmas, we refer to Keshavan et al. (2010).

In order to prove the lower bound, we start by noticing that

$$F(\mathbf{u}) \le \frac{1}{2}\|\mathcal{P}_E(Z)\|_F^2\,,$$

which is simply proved by using $S = \Sigma$ in Eq. (2). On the other hand, we have

$$\begin{aligned} F(\mathbf{x}) &= \frac{1}{2}\|\mathcal{P}_E(XSY^T - M - Z)\|_F^2 \\ &= \frac{1}{2}\|\mathcal{P}_E(Z)\|_F^2 + \frac{1}{2}\|\mathcal{P}_E(XSY^T - M)\|_F^2 - \langle \mathcal{P}_E(Z), (XSY^T - M)\rangle \qquad (17)\\ &\ge F(\mathbf{u}) + Cn\varepsilon\sqrt{\alpha}\, d_-(\mathbf{x},\mathbf{u})^2 - \sqrt{2r}\|Z^E\|_2\|XSY^T - M\|_F\,, \end{aligned}$$

where in the last step we used Lemma 7. Now by triangular inequality

$$\begin{aligned} \|XSY^T - M\|_F^2 &\le 3\|X(S-\Sigma)Y^T\|_F^2 + 3\|X\Sigma(Y-V)^T\|_F^2 + 3\|(X-U)\Sigma V^T\|_F^2 \\ &\le 3nm\|S-\Sigma\|_F^2 + 3n^2\alpha\Sigma_{\max}^2\Big(\frac{1}{m}\|X-U\|_F^2 + \frac{1}{n}\|Y-V\|_F^2\Big) \\ &\le Cn^2\alpha d_+(\mathbf{x},\mathbf{u})^2\,, \qquad (18) \end{aligned}$$

In order to prove the upper bound, we proceed as above to get

$$F(\mathbf{x}) \le \tfrac{1}{2}\|\mathcal{P}_E(Z)\|_F^2 + Cn\varepsilon\sqrt{\alpha}\Sigma_{\max}^2\, d(\mathbf{x},\mathbf{u})^2 + \sqrt{2r\alpha}\|Z^E\|_2 Cn d_+(\mathbf{x},\mathbf{u})\,.$$

Further, by replacing $\mathbf{x}$ with $\mathbf{u}$ in Eq. (17)

$$\begin{aligned} F(\mathbf{u}) &\ge \frac{1}{2}\|\mathcal{P}_E(Z)\|_F^2 - \langle \mathcal{P}_E(Z), (U(S-\Sigma)V^T)\rangle \\ &\ge \frac{1}{2}\|\mathcal{P}_E(Z)\|_F^2 - \sqrt{2r\alpha}\|Z^E\|_2 Cn d_+(\mathbf{x},\mathbf{u})\,. \end{aligned}$$

By taking the difference of these inequalities we get the desired upper bound. ■

Proof (Corollary 3.1) By putting together Eq. (8) and (9), and using the definitions of $d_+(\mathbf{x},\mathbf{u})$, $d_-(\mathbf{x},\mathbf{u})$, we get

$$\|S-\Sigma\|_F^2 \le \frac{C_1+C_2}{C_1}\Sigma_{\max}^2 d(\mathbf{x},\mathbf{u})^2 + \frac{(C_1+C_2)\sqrt{r}}{C_1\varepsilon}\|Z^E\|_2\sqrt{\Sigma_{\max}^2 d(\mathbf{x},\mathbf{u})^2 + \|S-\Sigma\|_F^2}\,.$$

Let $x \equiv \|S-\Sigma\|_F$, $a^2 \equiv \big((C_1+C_2)/C_1\big)\Sigma_{\max}^2 d(\mathbf{x},\mathbf{u})^2$, and $b \equiv \big((C_1+C_2)\sqrt{r}/C_1\varepsilon\big)\|Z^E\|_2$. The above inequality then takes the form

$$x^2 \le a^2 + b\sqrt{x^2+a^2} \le a^2 + ab + bx\,,$$

which implies our claim $x \le a+b$.

The singular value bounds (10) and (11) follow by triangular inequality. For instance

$$\sigma_{\min}(S) \geq \Sigma_{\min} - C\Sigma_{\max} d(\mathbf{x},\mathbf{u}) - C\frac{\sqrt{r}}{\varepsilon}\|Z^E\|_2 .$$

which implies the inequality (11) for $d(\mathbf{x},\mathbf{u}) \leq \delta = \Sigma_{\min}/C_0\Sigma_{\max}$ and C_0 large enough. An analogous argument proves Eq. (10). ■

3.4 Proof of Lemma 5

Without loss of generality we will assume $\delta \leq 1, C_2 \geq 1$ and

$$\frac{\sqrt{r}}{\varepsilon}\|Z^E\|_2 \leq \Sigma_{\min} , \tag{19}$$

because otherwise the lower bound (12) is trivial for all $d(\mathbf{x},\mathbf{u}) \leq \delta$.

Denote by $t \mapsto \mathbf{x}(t)$, $t \in [0,1]$, the geodesic on $\mathsf{M}(m,n)$ such that $\mathbf{x}(0) = \mathbf{u}$ and $\mathbf{x}(1) = \mathbf{x}$, parametrized proportionally to the arclength. Let $\widehat{\mathbf{w}} = \dot{\mathbf{x}}(1)$ be its final velocity, with $\widehat{\mathbf{w}} = (\widehat{W},\widehat{Q})$. Obviously $\widehat{\mathbf{w}} \in \mathsf{T}_{\mathbf{x}}$ (with $\mathsf{T}_{\mathbf{x}}$ the tangent space of $\mathsf{M}(m,n)$ at $\mathbf{x}$) and

$$\frac{1}{m}\|\widehat{W}\|^2 + \frac{1}{n}\|\widehat{Q}\|^2 = d(\mathbf{x},\mathbf{u})^2,$$

because $t \mapsto \mathbf{x}(t)$ is parametrized proportionally to the arclength.

Explicit expressions for $\widehat{\mathbf{w}}$ can be obtained in terms of $\mathbf{w} \equiv \dot{\mathbf{x}}(0) = (W,Q)$ (Keshavan et al., 2010). If we let $W = L\Theta R^T$ be the singular value decomposition of W, we obtain

$$\widehat{W} = -UR\Theta\sin\Theta R^T + L\Theta\cos\Theta R^T . \tag{20}$$

It was proved in Keshavan et al. (2010) that $\langle \operatorname{grad} G(\mathbf{x}), \widehat{\mathbf{w}}\rangle \geq 0$. It is therefore sufficient to lower bound the scalar product $\langle \operatorname{grad} F, \widehat{\mathbf{w}}\rangle$. By computing the gradient of F we get

$$\begin{aligned}
\langle \operatorname{grad} F(\mathbf{x}), \widehat{\mathbf{w}}\rangle &= \langle \mathcal{P}_E(XSY^T - N), (XS\widehat{Q}^T + \widehat{W}SY^T)\rangle \\
&= \langle \mathcal{P}_E(XSY^T - M), (XS\widehat{Q}^T + \widehat{W}SY^T)\rangle - \langle \mathcal{P}_E(Z), (XS\widehat{Q}^T + \widehat{W}SY^T)\rangle \\
&= \langle \operatorname{grad} F_0(\mathbf{x}), \widehat{\mathbf{w}}\rangle - \langle \mathcal{P}_E(Z), (XS\widehat{Q}^T + \widehat{W}SY^T)\rangle
\end{aligned} \tag{21}$$

where $F_0(\mathbf{x})$ is the cost function in absence of noise, namely

$$F_0(X,Y) = \min_{S\in\mathbb{R}^{r\times r}} \left\{ \frac{1}{2}\sum_{(i,j)\in E} \left((XSY^T)_{ij} - M_{ij}\right)^2 \right\} . \tag{22}$$

As proved in Keshavan et al. (2010),

$$\langle \operatorname{grad} F_0(\mathbf{x}), \widehat{\mathbf{w}}\rangle \geq Cn\varepsilon\sqrt{\alpha}\Sigma_{\min}^2 d(\mathbf{x},\mathbf{u})^2 \tag{23}$$

(see Lemma 9 in Appendix).

We are therefore left with the task of upper bounding $\langle \mathcal{P}_E(Z), (XS\widehat{Q}^T + \widehat{W}SY^T)\rangle$. Since $XS\widehat{Q}^T$ has rank at most r, we have

$$\langle \mathcal{P}_E(Z), XS\widehat{Q}^T\rangle \leq \sqrt{r}\,\|Z^E\|_2\,\|XS\widehat{Q}^T\|_F .$$

Since $X^T X = m\mathbf{I}$, we get

$$\begin{aligned} \|XS\widehat{Q}^T\|_F^2 &= m\mathrm{Tr}(S^T S\widehat{Q}^T\widehat{Q}) \le n\alpha\sigma_{\max}(S)^2\|\widehat{Q}\|_F^2 \\ &\le Cn^2\alpha\Big(\Sigma_{\max} + \frac{\sqrt{r}}{\varepsilon}\|Z^E\|_F\Big)^2 d(\mathbf{x},\mathbf{u})^2 \qquad (24) \\ &\le 4Cn^2\alpha\Sigma_{\max}^2\, d(\mathbf{x},\mathbf{u})^2\,, \end{aligned}$$

where, in inequality (24), we used Corollary 3.1 and in the last step, we used Eq. (19). Proceeding analogously for $\langle \mathcal{P}_E(Z), \widehat{W}SY^T\rangle$, we get

$$\langle \mathcal{P}_E(Z), (XS\widehat{Q}^T + \widehat{W}SY^T)\rangle \le C' n\Sigma_{\max}\sqrt{r\alpha}\,\|Z^E\|_2\, d(\mathbf{x},\mathbf{u})\,.$$

Together with Eq. (21) and (23) this implies

$$\langle \mathrm{grad}\, F(\mathbf{x}), \widehat{\mathbf{w}}\rangle \ge C_1 n\varepsilon\sqrt{\alpha}\Sigma_{\min}^2 d(\mathbf{x},\mathbf{u})\Big\{d(\mathbf{x},\mathbf{u}) - C_2\frac{\sqrt{r}\Sigma_{\max}}{\varepsilon\Sigma_{\min}}\frac{\|Z^E\|_2}{\Sigma_{\min}}\Big\}\,,$$

which implies Eq. (12) by Cauchy-Schwartz inequality.

4. Proof of Theorem 1.3

Proof (*Independent entries model*) We start with a claim that for any sampling set E, we have

$$\|\widetilde{Z}^E\|_2 \le \|Z^E\|_2\,.$$

To prove this claim, let x^* and y^* be m and n dimensional vectors, respectively, achieving the optimum in $\max_{\|x\|\le 1,\|y\|\le 1}\{x^T\widetilde{Z}^E y\}$, that is, such that $\|\widetilde{Z}^E\|_2 = x^{*T}\widetilde{Z}^E y^*$. Recall that, as a result of the trimming step, all the entries in trimmed rows and columns of $\widetilde{Z}^E$ are set to zero. Then, there is no gain in maximizing $x^T\widetilde{Z}^E y$ to have a non-zero entry x_i^* for i corresponding to the rows which are trimmed. Analogously, for j corresponding to the trimmed columns, we can assume without loss of generality that $y_j^* = 0$. From this observation, it follows that $x^{*T}\widetilde{Z}^E y^* = x^{*T}Z^E y^*$, since the trimmed matrix $\widetilde{Z}^E$ and the sample noise matrix Z^E only differ in the trimmed rows and columns. The claim follows from the fact that $x^{*T}Z^E y^* \le \|Z^E\|_2$, for any x^* and y^* with unit norm.

In what follows, we will first prove that $\|Z^E\|_2$ is bounded by the right-hand side of Eq. (4) for any range of $|E|$. Due to the above observation, this implies that $\|\widetilde{Z}^E\|_2$ is also bounded by $C\sigma\sqrt{\varepsilon\sqrt{\alpha}\log n}$, where $\varepsilon \equiv |E|/\sqrt{\alpha}n$. Further, we use the same analysis to prove a tighter bound in Eq. (5) when $|E| \ge n\log n$.

First, we want to show that $\|Z^E\|_2$ is bounded by $C\sigma\sqrt{\varepsilon\sqrt{\alpha}\log n}$, and Z_{ij}'s are i.i.d. random variables with zero mean and sub-Gaussian tail with parameter σ^2. The proof strategy is to show that $\mathbb{E}\big[\|Z^E\|_2\big]$ is bounded, using the result of Seginer (2000) on expected norm of random matrices, and use the fact that $\|\cdot\|_2$ is a Lipschitz continuous function of its arguments together with concentration inequality for Lipschitz functions on i.i.d. Gaussian random variables due to Talagrand (1996).

Note that $\|\cdot\|_2$ is a Lipschitz function with a Lipschitz constant 1. Indeed, for any M and M', $\big|\|M'\|_2 - \|M\|_2\big| \le \|M' - M\|_2 \le \|M' - M\|_F$, where the first inequality follows from triangular inequality and the second inequality follows from the fact that $\|\cdot\|_F^2$ is the sum of the squared singular values.

To bound the probability of large deviation, we use the result on concentration inequality for Lipschitz functions on i.i.d. sub-Gaussian random variables due to Talagrand (1996). For a 1-Lipschitz function $\|\cdot\|_2$ on $m \times n$ i.i.d. random variables Z^E_{ij} with zero mean, and sub-Gaussian tails with parameter σ^2,

$$\mathbb{P}\big(\|Z^E\|_2 - \mathbb{E}[\|Z^E\|_2] > t\big) \leq \exp\Big\{-\frac{t^2}{2\sigma^2}\Big\}. \tag{25}$$

Setting $t = \sqrt{8\sigma^2 \log n}$, this implies that $\|Z^E\|_2 \leq \mathbb{E}\big[\|Z\|_2\big] + \sqrt{8\sigma^2 \log n}$ with probability larger than $1 - 1/n^4$.

Now, we are left to bound the expectation $\mathbb{E}\big[\|Z^E\|_2\big]$. First, we symmetrize the possibly asymmetric random variables Z^E_{ij} to use the result of Seginer (2000) on expected norm of random matrices with symmetric random variables. Let Z'_{ij}'s be independent copies of Z_{ij}'s, and ξ_{ij}'s be independent Bernoulli random variables such that $\xi_{ij} = +1$ with probability $1/2$ and $\xi_{ij} = -1$ with probability $1/2$. Then, by convexity of $\mathbb{E}\big[\|Z^E - Z'^E\|_2 | Z'^E\big]$ and Jensen's inequality,

$$\mathbb{E}\big[\|Z^E\|_2\big] \leq \mathbb{E}\big[\|Z^E - Z'^E\|_2\big] = \mathbb{E}\big[\|(\xi_{ij}(Z^E_{ij} - Z'^E_{ij}))\|_2\big] \leq 2\mathbb{E}\big[\|(\xi_{ij} Z^E_{ij})\|_2\big],$$

where $(\xi_{ij} Z^E_{ij})$ denotes an $m \times n$ matrix with entry $\xi_{ij} Z^E_{ij}$ in position (i, j). Thus, it is enough to show that $\mathbb{E}\big[\|Z^E\|_2\big]$ is bounded by $C\sigma\sqrt{\varepsilon\sqrt{\alpha}\log n}$ in the case of symmetric random variables Z_{ij}'s.

To this end, we apply the following bound on expected norm of random matrices with i.i.d. symmetric random entries, proved by Seginer (2000, Theorem 1.1).

$$\mathbb{E}\big[\|Z^E\|_2\big] \leq C\Big(\mathbb{E}\big[\max_{i\in[m]} \|Z^E_{i\bullet}\|\big] + \mathbb{E}\big[\max_{j\in[n]} \|Z^E_{\bullet j}\|\big]\Big), \tag{26}$$

where $Z^E_{i\bullet}$ and $Z^E_{\bullet j}$ denote the ith row and jth column of A respectively. For any positive parameter β, which will be specified later, the following is true.

$$\mathbb{E}\big[\max_j \|Z^E_{\bullet j}\|^2\big] \leq \beta\sigma^2\varepsilon\sqrt{\alpha} + \int_0^\infty \mathbb{P}\big(\max_j \|Z^E_{\bullet j}\|^2 \geq \beta\sigma^2\varepsilon\sqrt{\alpha} + z\big)\,\mathrm{d}z. \tag{27}$$

To bound the second term, we can apply union bound on each of the n columns, and use the following bound on each column $\|Z^E_{\bullet j}\|^2$ resulting from concentration of measure inequality for the i.i.d. sub-Gaussian random matrix Z.

$$\mathbb{P}\Big(\sum_{k=1}^m (Z^E_{kj})^2 \geq \beta\sigma^2\varepsilon\sqrt{\alpha} + z\Big) \leq \exp\Big\{-\frac{3}{8}\Big((\beta-3)\varepsilon\sqrt{\alpha} + \frac{z}{\sigma^2}\Big)\Big\}. \tag{28}$$

To prove the above result, we apply Chernoff bound on the sum of independent random variables. Recall that $Z^E_{kj} = \tilde{\xi}_{kj} Z_{kj}$ where $\tilde{\xi}$'s are independent Bernoulli random variables such that $\tilde{\xi} = 1$ with probability $\varepsilon/\sqrt{mn}$ and zero with probability $1 - \varepsilon/\sqrt{mn}$. Then, for the choice of $\lambda = 3/8\sigma^2 < 1/2\sigma^2$,

$$\begin{aligned}
\mathbb{E}\Big[\exp\Big(\lambda \sum_{k=1}^m (\tilde{\xi}_{kj} Z_{kj})^2\Big)\Big] &= \Big(1 - \frac{\varepsilon}{\sqrt{mn}} + \frac{\varepsilon}{\sqrt{mn}}\mathbb{E}[e^{\lambda Z^2_{kj}}]\Big)^m \\
&\leq \Big(1 - \frac{\varepsilon}{\sqrt{mn}} + \frac{\varepsilon}{\sqrt{mn(1-2\sigma^2\lambda)}}\Big)^m \\
&= \exp\Big\{m\log\Big(1 + \frac{\varepsilon}{\sqrt{mn}}\Big)\Big\} \\
&\leq \exp\big\{\varepsilon\sqrt{\alpha}\big\},
\end{aligned}$$

where the first inequality follows from the definition of Z_{kj} as a zero mean random variable with sub-Gaussian tail, and the second inequality follows from $\log(1+x) \le x$. By applying Chernoff bound, Eq. (28) follows. Note that an analogous result holds for the Euclidean norm on the rows $\|Z^E_{i\bullet}\|^2$.

Substituting Eq. (28) and $\mathbb{P}\big(\max_j \|Z^E_{\bullet j}\|^2 \ge z\big) \le m\,\mathbb{P}\big(\|Z^E_{\bullet j}\|^2 \ge z\big)$ in Eq. (27), we get

$$\mathbb{E}\big[\max_j \|Z^E_{\bullet j}\|^2\big] \le \beta\sigma^2\varepsilon\sqrt{\alpha} + \frac{8\sigma^2 m}{3} e^{-\frac{3}{8}(\beta-3)\varepsilon\sqrt{\alpha}}\,. \tag{29}$$

The second term can be made arbitrarily small by taking $\beta = C\log n$ with large enough C. Since $\mathbb{E}\big[\max_j \|Z^E_{\bullet j}\|\big] \le \sqrt{\mathbb{E}\big[\max_j \|Z^E_{\bullet j}\|^2\big]}$, applying Eq. (29) with $\beta = C\log n$ in Eq. (26) gives

$$\mathbb{E}\big[\|Z^E\|_2\big] \le C\sigma\sqrt{\varepsilon\sqrt{\alpha}\log n}\,.$$

Together with Eq. (25), this proves the desired thesis for any sample size $|E|$.

In the case when $|E| \ge n\log n$, we can get a tighter bound by similar analysis. Since $\varepsilon \ge C'\log n$, for some constant C', the second term in Eq. (29) can be made arbitrarily small with a large constant β. Hence, applying Eq. (29) with $\beta = C$ in Eq. (26), we get

$$\mathbb{E}\big[\|Z^E\|_2\big] \le C\sigma\sqrt{\varepsilon\sqrt{\alpha}}\,.$$

Together with Eq. (25), this proves the desired thesis for $|E| \ge n\log n$.

■

Proof (*Worst Case Model*) Let D be the $m \times n$ all-ones matrix. Then for any matrix Z from the *worst case model*, we have $\|\widetilde{Z}^E\|_2 \le Z_{\max}\|\widetilde{D}^E\|_2$, since $x^T\widetilde{Z}^E y \le \sum_{i,j} Z_{\max}|x_i|\widetilde{D}^E_{ij}|y_j|$, which follows from the fact that Z_{ij}'s are uniformly bounded. Further, $\widetilde{D}^E$ is an adjacency matrix of a corresponding bipartite graph with bounded degrees. Then, for any choice of E the following is true for all positive integers k:

$$\|\widetilde{D}^E\|_2^{2k} \le \max_{x,\|x\|=1} \big|x^T((\widetilde{D}^E)^T\widetilde{D}^E)^k x\big| \le \mathrm{Tr}\big(((\widetilde{D}^E)^T\widetilde{D}^E)^k\big) \le n(2\varepsilon)^{2k}\,.$$

Now $\mathrm{Tr}\big(((\widetilde{D}^E)^T\widetilde{D}^E)^k\big)$ is the number of paths of length $2k$ on the bipartite graph with adjacency matrix $\widetilde{D}^E$, that begin and end at i for every $i \in [n]$. Since this graph has degree bounded by 2ε, we get

$$\|\widetilde{D}^E\|_2^{2k} \le n(2\varepsilon)^{2k}\,.$$

Taking k large, we get the desired thesis. ■

Acknowledgments

This work was partially supported by a Terman fellowship, the NSF CAREER award CCF-0743978 and the NSF grant DMS-0806211. SO was supported by a fellowship from the Samsung Scholarship Foundation.

Appendix A. Three Lemmas on the Noiseless Problem

Lemma 7 *There exists numerical constants C_0, C_1, C_2 such that the following happens. Assume $\varepsilon \geq C_0 \mu_0 r \sqrt{\alpha} \max\{\log n; \mu_0 r \sqrt{\alpha} (\Sigma_{\min}/\Sigma_{\max})^4\}$ and $\delta \leq \Sigma_{\min}/(C_0 \Sigma_{\max})$. Then,*

$$C_1 \sqrt{\alpha} \Sigma_{\min}^2 d(\mathbf{x},\mathbf{u})^2 + C_1 \sqrt{\alpha} \|S_0 - \Sigma\|_F^2 \leq \frac{1}{n\varepsilon} F_0(\mathbf{x}) \leq C_2 \sqrt{\alpha} \Sigma_{\max}^2 d(\mathbf{x},\mathbf{u})^2 ,$$

for all $\mathbf{x} \in \mathsf{M}(m,n) \cap \mathcal{K}(4\mu_0)$ such that $d(\mathbf{x},\mathbf{u}) \leq \delta$, with probability at least $1 - 1/n^4$. Here $S_0 \in \mathbb{R}^{r \times r}$ is the matrix realizing the minimum in Eq. (22).

Lemma 8 *There exists numerical constants C_0 and C such that the following happens. Assume $\varepsilon \geq C_0 \mu_0 r \sqrt{\alpha} (\Sigma_{\max}/\Sigma_{\min})^2 \max\{\log n; \mu_0 r \sqrt{\alpha} (\Sigma_{\max}/\Sigma_{\min})^4\}$ and $\delta \leq \Sigma_{min}/(C_0 \Sigma_{\max})$. Then*

$$\|\operatorname{grad} \widetilde{F}_0(\mathbf{x})\|^2 \geq C n \varepsilon^2 \Sigma_{\min}^4 d(\mathbf{x},\mathbf{u})^2 ,$$

for all $\mathbf{x} \in \mathsf{M}(m,n) \cap \mathcal{K}(4\mu_0)$ such that $d(\mathbf{x},\mathbf{u}) \leq \delta$, with probability at least $1 - 1/n^4$.

Lemma 9 *Define $\widehat{\mathbf{w}}$ as in Eq. (20). Then there exists numerical constants C_0 and C such that the following happens. Under the hypothesis of Lemma 8*

$$\langle \operatorname{grad} F_0(\mathbf{x}), \widehat{\mathbf{w}} \rangle \geq C n \varepsilon \sqrt{\alpha} \Sigma_{\min}^2 d(\mathbf{x},\mathbf{u})^2 ,$$

for all $\mathbf{x} \in \mathsf{M}(m,n) \cap \mathcal{K}(4\mu_0)$ such that $d(\mathbf{x},\mathbf{u}) \leq \delta$, with probability at least $1 - 1/n^4$.

References

P.-A. Absil, R. Mahony, and R. Sepulchrer. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

D. Achlioptas and F. McSherry. Fast computation of low-rank matrix approximations. *J. ACM*, 54 (2):9, 2007.

J-F Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. `arXiv:0810.3286`, 2008.

E. J. Candès and Y. Plan. Matrix completion with noise. `arXiv:0903.3131`, 2009.

E. J. Candès and B. Recht. Exact matrix completion via convex optimization. `arxiv:0805.4471`, 2008.

E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. `arXiv:0903.1476`, 2009.

A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matr. Anal. Appl.*, 20:303–353, 1999.

M. Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University, 2002.

A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004. ISSN 0004-5411.

R. H. Keshavan and S. Oh. Optspace: A gradient descent algorithm on the grassman manifold for matrix completion. arXiv:0910.5260, 2009.

R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Trans. Inform. Theory*, 56(6):2980–2998, June 2010.

K. Lee and Y. Bresler. Admira: Atomic decomposition for minimum rank approximation. arXiv:0905.0044, 2009.

S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. arXiv:0905.1643, 2009.

B. Recht, M. Fazel, and P. Parrilo. Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization. arxiv:0706.4138, 2007.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

R. Salakhutdinov and N. Srebro. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. arXiv:1002.2780, 2010.

R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine Learning*, volume 24, pages 791–798, 2007.

Y. Seginer. The expected norm of random matrices. *Comb. Probab. Comput.*, 9:149–166, March 2000. ISSN 0963-5483. doi: 10.1017/S096354830000420X. URL http://portal.acm.org/citation.cfm?id=971471.971475.

N. Srebro and T. S. Jaakkola. Weighted low-rank approximations. In *In 20th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

N. Srebro, J. D. M. Rennie, and T. S. Jaakola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.

M. Talagrand. A new look at independence. *The Annals of Probability*, 24(1):1–34, 1996. ISSN 00911798. URL http://www.jstor.org/stable/2244830.

K. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. http://www.math.nus.edu.sg/∼matys, 2009.

Journal of Machine Learning Research 11 (2010) 2079-2107 Submitted 10/09; Revised 3/10; Published 7/10

On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation

Gavin C. Cawley GCC@CMP.UEA.AC.UK
Nicola L. C. Talbot NLCT@CMP.UEA.AC.UK
School of Computing Sciences
University of East Anglia
Norwich, United Kingdom NR4 7TJ

Editor: Isabelle Guyon

Abstract

Model selection strategies for machine learning algorithms typically involve the numerical optimisation of an appropriate model selection criterion, often based on an estimator of generalisation performance, such as k-fold cross-validation. The error of such an estimator can be broken down into bias and variance components. While unbiasedness is often cited as a beneficial quality of a model selection criterion, we demonstrate that a low variance is at least as important, as a non-negligible variance introduces the potential for over-fitting in model selection as well as in training the model. While this observation is in hindsight perhaps rather obvious, the degradation in performance due to over-fitting the model selection criterion can be surprisingly large, an observation that appears to have received little attention in the machine learning literature to date. In this paper, we show that the effects of this form of over-fitting are often of comparable magnitude to differences in performance between learning algorithms, and thus cannot be ignored in empirical evaluation. Furthermore, we show that some common performance evaluation practices are susceptible to a form of selection bias as a result of this form of over-fitting and hence are unreliable. We discuss methods to avoid over-fitting in model selection and subsequent selection bias in performance evaluation, which we hope will be incorporated into best practice. While this study concentrates on cross-validation based model selection, the findings are quite general and apply to any model selection practice involving the optimisation of a model selection criterion evaluated over a finite sample of data, including maximisation of the Bayesian evidence and optimisation of performance bounds.

Keywords: model selection, performance evaluation, bias-variance trade-off, selection bias, over-fitting

1. Introduction

This paper is concerned with two closely related topics that form core components of best practice in both the real world application of machine learning methods and the development of novel machine learning algorithms, namely model selection and performance evaluation. The majority of machine learning algorithms are based on some form of multi-level inference, where the model is defined by a set of model parameters and also a set of hyper-parameters (Guyon et al., 2009), for example in kernel learning methods the parameters correspond to the coefficients of the kernel expansion and the hyper-parameters include the regularisation parameter, the choice of kernel function and any associated kernel parameters. This division into parameters and hyper-parameters is typically

performed for computational convenience; for instance in the case of kernel machines, for fixed values of the hyper-parameters, the parameters are normally given by the solution of a convex optimisation problem for which efficient algorithms are available. Thus it makes sense to take advantage of this structure and fit the model iteratively using a pair of nested loops, with the hyper-parameters adjusted to optimise a model selection criterion in the outer loop (model selection) and the parameters set to optimise a training criterion in the inner loop (model fitting/training). In our previous study (Cawley and Talbot, 2007), we noted that the variance of the model selection criterion admitted the possibility of over-fitting during model selection as well as the more familiar form of over-fitting that occurs during training and demonstrated that this could be ameliorated to some extent by regularisation of the model selection criterion. The first part of this paper discusses the problem of over-fitting in model selection in more detail, providing illustrative examples, and describes how to avoid this form of over-fitting in order to gain the best attainable performance, desirable in practical applications, and required for fair comparison of machine learning algorithms.

Unbiased and robust[1] performance evaluation is undoubtedly the cornerstone of machine learning research; without a reliable indication of the relative performance of competing algorithms, across a wide range of learning tasks, we cannot have the clear picture of the strengths and weaknesses of current approaches required to set the direction for future research. This topic is considered in the second part of the paper, specifically focusing on the undesirable optimistic bias that can arise due to over-fitting in model selection. This phenomenon is essentially analogous to the selection bias observed by Ambroise and McLachlan (2002) in microarray classification, due to feature selection prior to performance evaluation, and shares a similar solution. We show that some, apparently quite benign, performance evaluation protocols in common use by the machine learning community are susceptible to this form of bias, and thus potentially give spurious results. In order to avoid this bias, model selection must be treated as an integral part of the model fitting process and performed afresh every time a model is fitted to a new sample of data. Furthermore, as the differences in performance due to model selection are shown to be often of comparable magnitude to the difference in performance between learning algorithms, it seems no longer meaningful to evaluate the performance of machine learning algorithms in isolation, and we should instead compare learning algorithm/model selection procedure combinations. However, this means that robust unbiased performance evaluation is likely to require more rigorous and computationally intensive protocols, such a nested cross-validation or "double cross" (Stone, 1974).

None of the methods or algorithms discussed in this paper are new; the novel contribution of this work is an empirical demonstration that over-fitting at the second level of inference (i.e., model selection) can have a very substantial deleterious effect on the generalisation performance of state-of-the-art machine learning algorithms. Furthermore the demonstration that this can lead to a misleading optimistic bias in performance evaluation using evaluation protocols in common use in the machine learning community is also novel. The paper is intended to be of some tutorial value in promoting best practice in model selection and performance evaluation, however we also hope that the observation that over-fitting in model selection is a significant problem will encourage much needed algorithmic and theoretical development in this area.

The remainder of the paper is structured as follows: Section 2 provides a brief overview of the kernel ridge regression classifier used as the base classifier for the majority of the experimental work

1. The term "robust" is used here to imply insensitivity to irrelevant experimental factors, such as the sampling and partitioning of the data to form training, validation and test sets; this is normally achieved by computationally expensive resampling schemes, for example, cross-validation (Stone, 1974) and the bootstrap (Efron and Tibshirani, 1994).

and Section 3 describes the data sets used. Section 4 demonstrates the importance of the variance of the model selection criterion, as it can lead to over-fitting in model selection, resulting in poor generalisation performance. A number of methods to avoid over-fitting in model selection are also discussed. Section 5 shows that over-fitting in model selection can result in biased performance evaluation if model selection is not viewed as an integral part of the modelling procedure. Two apparently benign and widely used performance evaluation protocols are shown to be affected by this problem. Finally, the work is summarised in Section 6.

2. Kernel Ridge Regression

In this section, we provide a brief overview of the Kernel Ridge Regression (KRR) classifier (Saunders et al., 1998), also known as the Least-Squares Support Vector Machine (Suykens et al., 2002), Regularised Least Squares (Rifkin and Lippert, 2007), Regularisation Network (Poggio and Girosi, 1990) etc., used as the base classifier in most of the empirical demonstrations in the sequel. Assume we are given labeled training data, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{\ell}$, where $x_i \in \mathcal{X} \subset \mathbb{R}^d$ is a vector of input features describing the i^{th} example and $y_i \in \{-1, +1\}$ is an indicator variable such that $y_i = +1$ if the i^{th} example is drawn from the positive class, $\mathcal{C}^+$, and $y_i = -1$ if from the negative class, $\mathcal{C}^-$. Further let us assume there are ℓ^+ positive examples and $\ell^- = \ell - \ell^+$ negative examples. The Kernel Ridge Regression classifier aims to construct a linear model $f(x) = w \cdot \phi(x) + b$ in a fixed feature space, $\phi : \mathcal{X} \to \mathcal{F}$, that is able to distinguish between examples drawn from $\mathcal{C}^-$ and $\mathcal{C}^+$, such that

$$x \in \begin{cases} \mathcal{C}^+ & \text{if } f(x) \geq 0 \\ \mathcal{C}^- & \text{otherwise} \end{cases}.$$

However, rather than specifying the feature space, $\mathcal{F}$, directly, it is induced by a kernel function, $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, giving the inner product between the images of vectors in the feature space, $\mathcal{F}$, that is, $\mathcal{K}(x, x') = \phi(x) \cdot \phi(x')$. A common kernel function, used throughout this study, is the Gaussian radial basis function (RBF) kernel

$$\mathcal{K}(x, x') = \exp\left\{-\eta \|x - x'\|^2\right\}, \tag{1}$$

where η is a kernel parameter controlling the sensitivity of the kernel function. However, the interpretation of the kernel function as evaluating the inner product between points in an implied feature space is valid for any kernel for which the kernel matrix $K = [k_{ij} = \mathcal{K}(x_i, x_j)]_{i,j=1}^{\ell}$ is positive definite (Mercer, 1909), such that

$$a^T K a > 0, \qquad \forall\, a \neq 0.$$

The model parameters (w, b) are given by the minimum of a regularised (Tikhonov and Arsenin, 1977) least-squares loss function,

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + \frac{1}{2\lambda}\sum_{i=1}^{\ell}\left[y_i - w \cdot \phi(x_i) - b\right]^2, \tag{2}$$

where λ is a regularisation parameter controlling the bias-variance trade-off (Geman et al., 1992). The accuracy of the kernel machine on test data is critically dependent on the choice of good values for the *hyper-parameters*, in this case λ and η. The search for the optimal values for such hyper-parameters is a process known as *model selection*. The representer theorem (Kimeldorf and Wahba,

1971) states that the solution to this optimisation problem can be written as an expansion of the form

$$w = \sum_{i=1}^{\ell} \alpha_i \phi(x_i) \quad \Longrightarrow \quad f(x) = \sum_{i=1}^{\ell} \alpha_i \mathcal{K}(x_i, x) + b.$$

The dual parameters of the kernel machine, α, are then given by the solution of a system of linear equations,

$$\begin{bmatrix} K + \lambda I & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}. \tag{3}$$

where $y = (y_1, y_2, \ldots, y_\ell)^T$, which can be solved efficiently via Cholesky factorisation of $K + \lambda I$, with a computational complexity of $O(\ell^3)$ operations (Suykens et al., 2002). The simplicity and efficiency of the kernel ridge regression classifier makes it an ideal candidate for relatively small-scale empirical investigations of practical issues, such as model selection.

2.1 Efficient Leave-One-Out Cross-Validation

Cross-validation (e.g., Stone, 1974) provides a simple and effective method for both model selection and performance evaluation, widely employed by the machine learning community. Under k-fold cross-validation the data are randomly partitioned to form k disjoint subsets of approximately equal size. In the i^{th} fold of the cross-validation procedure, the i^{th} subset is used to estimate the generalisation performance of a model trained on the remaining $k-1$ subsets. The average of the generalisation performance observed over all k folds provides an estimate (with a slightly pessimistic bias) of the generalisation performance of a model trained on the entire sample. The most extreme form of cross-validation, in which each subset contains only a single pattern is known as leave-one-out cross-validation (Lachenbruch and Mickey, 1968; Luntz and Brailovsky, 1969). An attractive feature of kernel ridge regression is that it is possible to perform leave-one-out cross-validation in closed form, with minimal cost as a by-product of the training algorithm (Cawley and Talbot, 2003). Let C represent the matrix on the left hand side of (3), then the residual error for the i^{th} training pattern in the i^{th} fold of the leave-one-out process is given by,

$$r_i^{(-i)} = y_i - \hat{y}_i^{(-i)} = \frac{\alpha_i}{C_{ii}^{-1}},$$

where $\hat{y}_i^{(-j)}$ is the output of the kernel ridge regression machine for the i^{th} observation in the j^{th} fold of the leave-one-out procedure and C_{ii}^{-1} is the i^{th} element of the principal diagonal of the inverse of the matrix C. Similar methods have been used in least-squares linear regression for many years, (e.g., Stone, 1974; Weisberg, 1985). While the optimal model parameters of the kernel machine are given by the solution of a simple system of linear equations, (3), some form of model selection is required to determine good values for the *hyper-parameters*, $\theta = (\lambda, \eta)$, in order to maximise generalisation performance. The analytic leave-one-out cross-validation procedure described here can easily be adapted to form the basis of an efficient model selection strategy (cf. Chapelle et al., 2002; Cawley and Talbot, 2003; Bo et al., 2006). In order to obtain a continuous model selection criterion, we adopt Allen's Predicted REsidual Sum-of-Squares (PRESS) statistic (Allen, 1974),

$$\text{PRESS}(\theta) = \sum_{i=1}^{\ell} \left[r_i^{(-i)} \right]^2.$$

The PRESS criterion can be optimised efficiently using scaled conjugate gradient descent (Williams, 1991) or Nelder-Mead simplex (Nelder and Mead, 1965) procedures. For full details of the training and model selection procedures for the kernel ridge regression classifier, see Cawley (2006). A public domain MATLAB implementation of the kernel ridge regression classifier, including automated model selection, is provided by the Generalised Kernel Machine (GKM) (Cawley et al., 2007) toolbox.[2]

3. Data Sets used in Empirical Demonstrations

In this section, we describe the benchmark data sets used in this study to illustrate the problem of over-fitting in model selection and to demonstrate the bias this can introduce into performance evaluation.

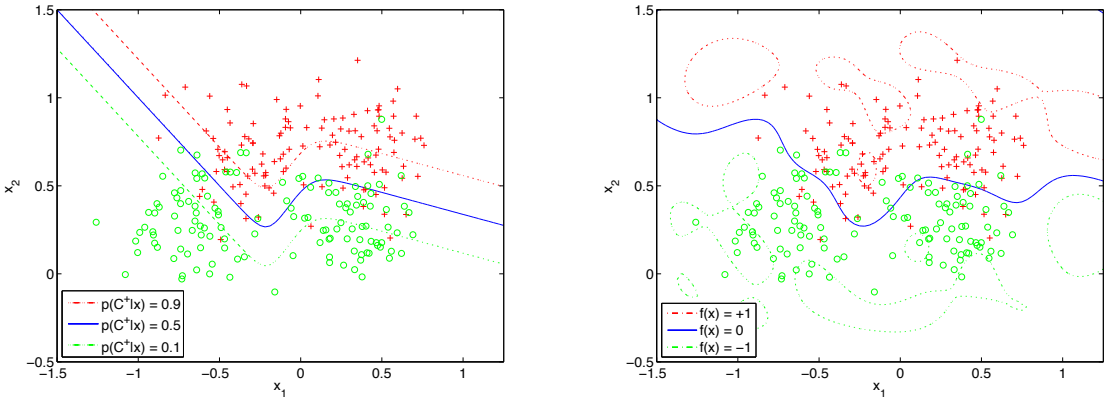
3.1 A Synthetic Benchmark

A synthetic benchmark, based on that introduced by Ripley (1996), is used widely in the next section to illustrate the nature of over-fitting in model selection. The data are drawn from four spherical bivariate Gaussian distributions, with equal probability. All four Gaussians have a common variance, $\sigma^2 = 0.04$. Patterns belonging to the positive classes are drawn from Gaussians centered on $[+0.4, +0.7]$ and $[-0.3, +0.7]$; the negative patterns are drawn from Gaussians centered on $[-0.7, +0.3]$ and $[+0.3, +0.3]$. Figure 1 shows a realisation of the synthetic benchmark, consisting of 256 patterns, showing the Bayes-optimal decision boundary and contours representing an a-posteriori probability of belonging to the positive class of 0.1 and 0.9. The Bayes error for this benchmark is approximately 12.38%. This benchmark is useful firstly as the Bayes optimal decision boundary is known, but also because it provides an inexhaustible supply of data, allowing the numerical approximation of various expectations.

3.2 A Suite of Benchmarks for Robust Performance Evaluation

In addition to illustrating the nature of over-fitting in model selection, we need to demonstrate that it is a serious concern in practical applications and show that it can result in biased performance evaluation if not taken into consideration. Table 1 gives the details of a suite of thirteen benchmark data sets, introduced by Rätsch et al. (2001). Each benchmark is based on a data set from the UCI machine learning repository, augmented by a set of 100 pre-defined partitions to form multiple realisations of the training and test sets (20 in the case of the larger `image` and `splice` data sets). The use of multiple benchmarks means that the evaluation is more robust as the selection of data sets that provide a good match to the inductive bias of a particular classifier becomes less likely. Likewise, the use of multiple partitions provides robustness against sensitivity to the sampling of data to form training and test sets. Results on this suite of benchmarks thus provides a reasonable indication of the magnitude of the effects of over-fitting in model selection that we might expect to see in practice.

2. Toolbox can be found at `http://theoval.cmp.uea.ac.uk/~gcc/projects/gkm`.



(a) (b)

Figure 1: Realisation of the Synthetic benchmark data set, with Bayes optimal decision boundary (a) and kernel ridge regression classifier with an automatic relevance determination (ARD) kernel where the hyper-parameters are tuned so as to minimise the true test MSE (b).

Data Set	Training Patterns	Testing Patterns	Number of Replications	Input Features
banana	400	4900	100	2
breast cancer	200	77	100	9
diabetis	468	300	100	8
flare solar	666	400	100	9
german	700	300	100	20
heart	170	100	100	13
image	1300	1010	20	18
ringnorm	400	7000	100	20
splice	1000	2175	20	60
thyroid	140	75	100	5
titanic	150	2051	100	3
twonorm	400	7000	100	20
waveform	400	4600	100	21

Table 1: Details of data sets used in empirical comparison.

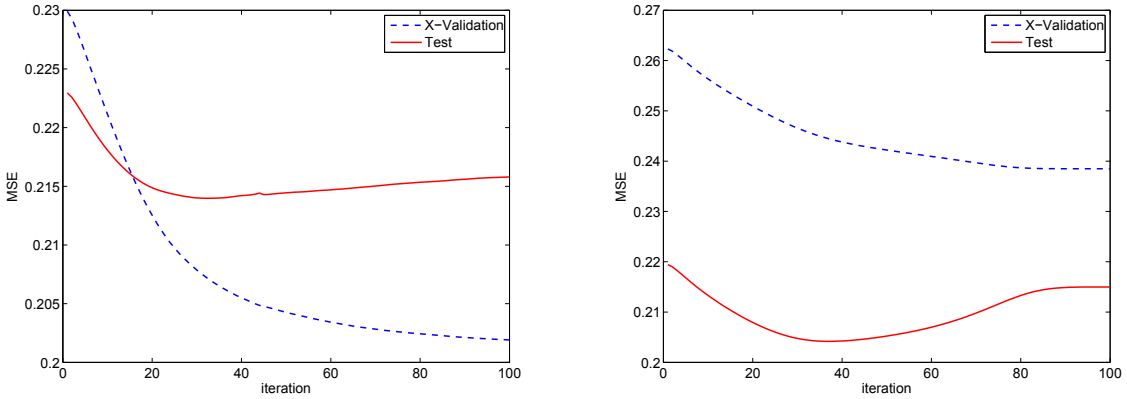
4. Over-fitting in Model Selection

We begin by demonstrating that it is possible to over-fit a model selection criterion based on a finite sample of data, using the synthetic benchmark problem, where ground truth is available. Here we use "over-fitting in model selection" to mean minimisation of the model selection criterion beyond the point at which generalisation performance ceases to improve and subsequently begins to decline.

Figure 1 (b) shows the output of a kernel ridge regression classifier for the synthetic problem, with the Automatic Relevance Determination (ARD) variant of the Gaussian radial basis function kernel,

$$\mathcal{K}(x,x') = \exp\left\{-\sum_{i=1}^{d} \eta_i (x_i - x_i')^2\right\},$$

which has a separate scaling parameter, η_i, for each feature. A much larger training set of 4096 samples was used, and the hyper-parameters were tuned to minimise the true test mean squared errors (MSE). The performance of this model achieved an error rate of 12.50%, which suggests that a model of this form is capable of approaching the Bayes error rate for this problem, at least in principle, and so there is little concern of model mis-specification.



(a) (b)

Figure 2: Evolution of the expected four-fold cross-validation and true test mean squared error as a function of the number of iterations (optimisation steps in the minimisation of the model selection criterion) of the model selection process, for a kernel ridge regression classifier trained on the `synthetic` benchmark data set (a) and (b) the evolution of those statistics for a particular realisation of the data set.

A further one thousand independent realisations of this benchmark were generated, each consisting of 64 samples. A kernel ridge regression classifier, based on the ARD kernel, was constructed for each realisation, with the hyper-parameters tuned so as to minimise a four-fold cross-validation estimate of the mean squared error. The true generalisation performance of each model was estimated numerically using the underlying generative model of the data set. Figure 2 (a) shows the expected true test and cross-validation estimates of the mean squared error averaged over all 1000 realisations of the benchmark. As would be expected, the cross-validation estimate of the mean squared error, forming the model selection criterion, is monotonically decreasing. However, the expected value of the true test MSE initially shows a decrease, as the hyper-parameters are modified in a manner that provides genuine improvements in generalisation, but after a relatively short time (approximately 30–40 iterations), the test error begins to climb slowly once more as the hyper-parameters are tuned in ways that exploit the meaningless statistical peculiarities of the sample. This produces a close analog of the classic plot used to illustrate the nature of over-fitting in training, for example, Figure

9.7 of the book by Bishop (1995). Figure 2 (b) shows the same statistics for one particular realisation of the data, demonstrating that the over-fitting can in some cases be quite substantial, clearly in this case some form of early-stopping in the model selection process would have resulted in improved generalisation. Having demonstrated that the classic signature of over-fitting during training is also apparent in the evolution of cross-validation and test errors during model selection, we discuss in the next section the origin of this form of over-fitting in terms of the *bias* and *variance* of the model selection criterion.

4.1 Bias and Variance in Model Selection

Model selection criteria are generally based on an estimator of generalisation performance evaluated over a finite sample of data, this includes resampling methods, such as split sample estimators, cross-validation (Stone, 1974) and bootstrap methods (Efron and Tibshirani, 1994), but also more loosely, the Bayesian evidence (MacKay, 1992; Rasmussen and Williams, 2006) and theoretical performance bounds such as the radius-margin bound (Vapnik, 1998). The error of an estimator can be decomposed into two components, *bias* and *variance*. Let $G(\theta)$ represent the true generalisation performance of a model with hyper-parameters θ, and $g(\theta;\mathcal{D})$ be an estimate of generalisation performance evaluated over a finite sample, $\mathcal{D}$, of n patterns. The expected squared error of the estimator can then be written in the form (Geman et al., 1992; Duda et al., 2001),

$$E_{\mathcal{D}}\left\{[g(\theta;\mathcal{D})-G(\theta)]^2\right\} = [E_{\mathcal{D}}\{g(\theta;\mathcal{D})-G(\theta)\}]^2 + E_{\mathcal{D}}\left\{\left[g(\theta;\mathcal{D})-E_{\mathcal{D}'}\left\{g(\theta;\mathcal{D}')\right\}\right]^2\right\},$$

where $E_{\mathcal{D}}\{\cdot\}$ represents an expectation evaluated over independent samples, $\mathcal{D}$, of size n. The first term, the squared *bias*, represents the difference between the expected value of the estimator and the unknown value of the true generalisation error. The second term, known as the *variance*, reflects the variability of the estimator around its expected value due to the sampling of the data $\mathcal{D}$ on which it is evaluated. Clearly if the expected squared error is low, we may reasonably expect $g(\cdot)$ to perform well as a model selection criterion. However, in practice, the expected squared error may be significant, in which case, it is interesting to ask whether the bias or the variance component is of greatest importance in reliably achieving optimal generalisation.

It is straightforward to demonstrate that leave-one-out cross-validation provides an almost unbiased estimate of the true generalisation performance (Luntz and Brailovsky, 1969), and this is often cited as being an advantageous property of the leave-one-out estimator in the setting of model selection (e.g., Vapnik, 1998; Chapelle et al., 2002). However, for the purpose of model selection, rather than performance evaluation, unbiasedness *per se* is relatively unimportant, instead the primary requirement is merely for the minimum of the model selection criterion to provide a reliable indication of the minimum of the true test error in hyper-parameter space. This point is illustrated in Figure 3, which shows a hypothetical example of a model selection criterion that is unbiased (by construction) (a) and another that is clearly biased (b). Unbiasedness provides the assurance that the minimum of the expected value of the model selection criterion, $E_{\mathcal{D}}\{g(\theta;\mathcal{D})\}$ coincides with the minimum of the test error, $G(\theta)$. However, in practice, we have only a finite sample of data, $\mathcal{D}_i$, over which to evaluate the model selection criterion, and so it is the minimum of $g(\theta;\mathcal{D}_i)$ that is of interest. In Figure 3 (a), it can be seen that while the estimator is unbiased, it has a high variance, and so there is a large spread in the values of θ at which the minimum occurs for different samples of data, and so $g(\theta;\mathcal{D}_i)$ is likely to provide a poor model selection criterion in practice. On the other hand, Figure 3 (b) shows a criterion with lower variance, and hence is the better model selection

criterion, despite being biased, as the minima of $g'(\theta;\mathcal{D}_i)$ for individual samples lie much closer to the minimum of the true test error. This demonstrates that while unbiasedness is reassuring, as it means that the form of the model selection criterion is correct *on average*, the variance of the criterion is also vitally important as it is this that ensures that the minimum of the selection criterion evaluated on a particular sample will provide good generalisation.

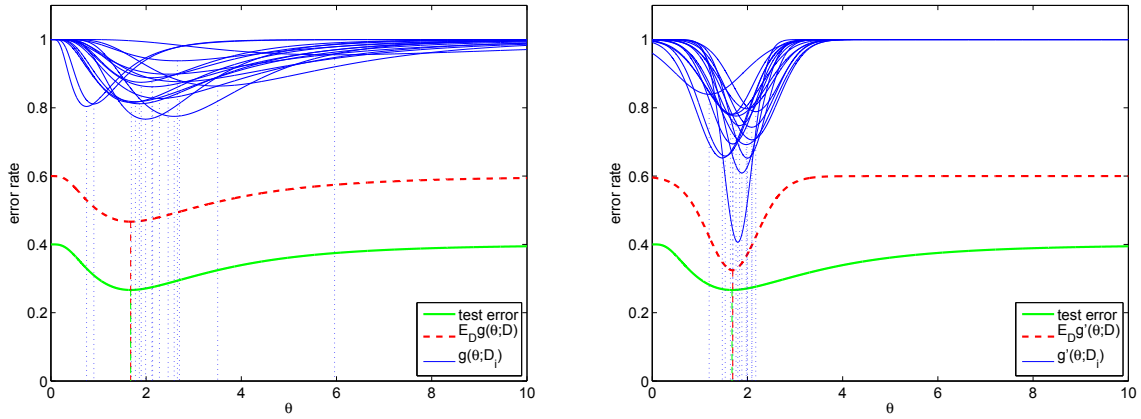


Figure 3: Hypothetical example of an unbiased (a) and a biased (b) model selection criterion. Note that the biased model selection criterion (b) is likely to provide the more effective model selection criterion as it has a lower variance, even though it is significantly biased. For clarity, the true error rate and the expected value of the model selection criteria are shown with vertical displacements of -0.6 and -0.4 respectively.

4.2 The Effects of Over-fitting in Model Selection

In this section, we investigate the effect of the variance of the model selection criterion using a more realistic example, again based on the `synthetic` benchmark, where the underlying generative model is known and so we are able to evaluate the true test error. It is demonstrated that over-fitting in model selection can cause both under-fitting and over-fitting of the training sample. A fixed training set of 256 patterns is generated and used to train a kernel ridge regression classifier, using the simple RBF kernel (1), with hyper-parameter settings defining a fine grid spanning reasonable values of the regularisation and kernel parameters, λ and η respectively. The smoothed error rate (Bo et al., 2006),

$$\mathrm{SER}(\theta) = \frac{1}{2n}\sum_{i=1}^{n}\left[1 - y_i \tanh\left\{\gamma f(x_i)\right\}\right]$$

is used as the statistic of interest, in order to improve the clarity of the figures, where γ is a parameter controlling the amount of smoothing applied ($\gamma = 8$ is used throughout, however the precise value is not critical). Figure 4 (a) shows the true test smoothed error rate as a function of the hyper-parameters. As these are both scale parameters, a logarithmic representation is used for both axes. The true test smoothed error rate is an approximately unimodal function of the hyper-parameters,

with a single distinct minimum, indicating the hyper-parameter settings giving optimal generalisation.

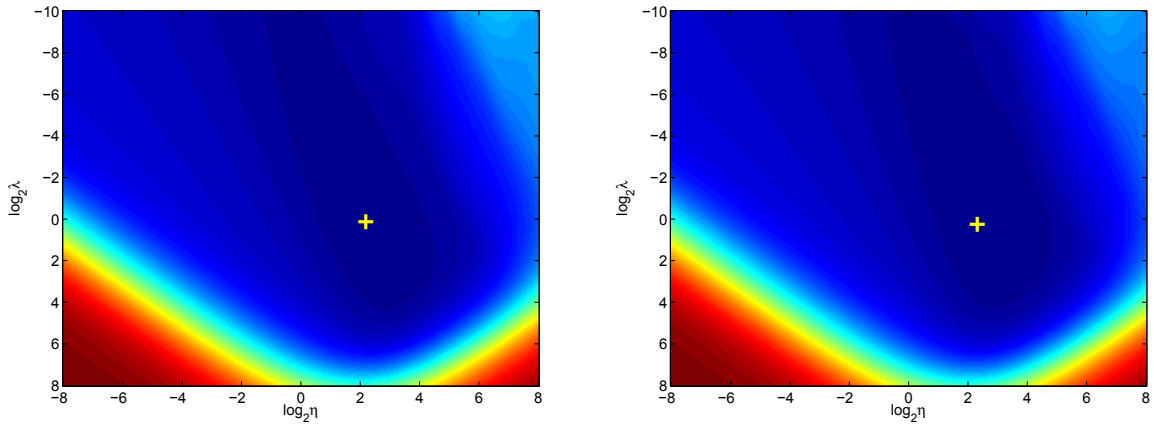


Figure 4: Plot of the true test smoothed error rate (a) and mean smoothed error rate over 100 random validation sets of 64 samples (b), for a kernel ridge regression classifier as a function of the hyper-parameters. In each case, the minimum is shown by a yellow cross, $+$.

In practical applications, however, the true test error is generally unknown, and so we must rely on an estimator of some sort. The simplest estimator for use in model selection is the error computed over an independent validation set, that is, the split-sample estimator. It seems entirely reasonable to expect the split-sample estimator to be unbiased. Figure 4 (b) shows a plot of the mean smoothed error rate using the split-sample estimator, over 100 random validation sets, each of which consists of 64 patterns. Note that the same fixed training set is used in each case. This plot is very similar to the true smoothed error, shown in Figure 4 (a), demonstrating that the split sample estimator is indeed approximately unbiased.

While the split-sample estimator is unbiased, it may have a high variance, especially as in this case the validation set is (intentionally) relatively small. Figure 5 shows plots of the split-sample estimate of the smoothed error rate for six selected realisations of a validation set of 64 patterns. Clearly, the split-sample error estimate is no longer as smooth, or indeed unimodal. More importantly, the hyper-parameter values selected by minimising the validation set error, and therefore the true generalisation performance, depends on the particular sample of data used to form the validation set. Figure 6 shows that the variance of the split-sample estimator can result in models ranging from severely under-fit (a) to severely over-fit (f), with variations in between these extremes.

Figure 7 (a) shows a scatter plot of the validation set and true error rates for kernel ridge regression classifiers for the synthetic benchmark, with split-sample based model selection using 100 random realisations of the validation set. Clearly, the split-sample based model selection procedure normally performs well. However, there is also significant variation in performance with different samples forming the validation set. We can also see that the validation set error is strongly biased, having been directly minimised during model selection, and (of course) should not be used for performance estimation.

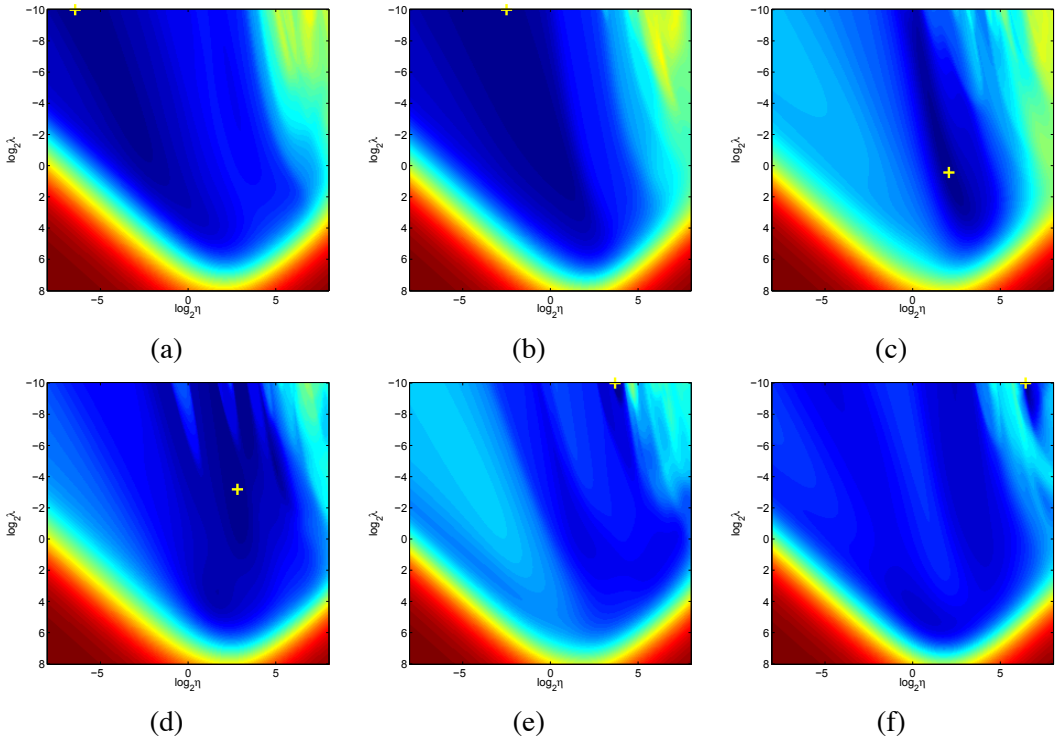


Figure 5: Contour plot of the split-sample estimate of the smoothed error rate for a kernel ridge regression machine as a function of the hyper-parameters, for six random realisations of the validation set. The minimum is shown by a cross, $+$.

Note that in this section we have deliberately employed a split-sample based model selection strategy with a relatively high variance, due to the limited size of the validation set. A straightforward way to reduce the variance of the model selection criterion is simply to increase the size of the validation sample over which it is evaluated. Figure 8 shows the optimal hyper-parameter settings obtained using 100 realisations of validation sets of 64 (a) and 256 (b) samples. It can be clearly seen that the use of a larger validation set has resulted in a tighter clustering of hyper-parameter values around the true optimum, note also that the hyper-parameters are concentrated along the bottom of a broad valley in hyper-parameter space, so even when the selected values are different from the optimal value, they still lie in positions giving good generalisation. This is further illustrated in Figure 7 (b), where the true smoothed error rates are much more tightly clustered, with fewer outliers, for the larger validation sets than obtained using smaller validation sets, shown in Figure 7 (a).

The variation in performance for different realisations of the benchmark suggests that evaluation of machine learning algorithms should always involve multiple partitions of the data to form training/validation and test sets, as the sampling of data for a single partition of the data might arbitrarily favour one classifier over another. This is illustrated in Figure 9, which shows the test error rates for Gaussian Process and Kernel Logistic Regression classifiers (GPC and KLR respectively), for 100

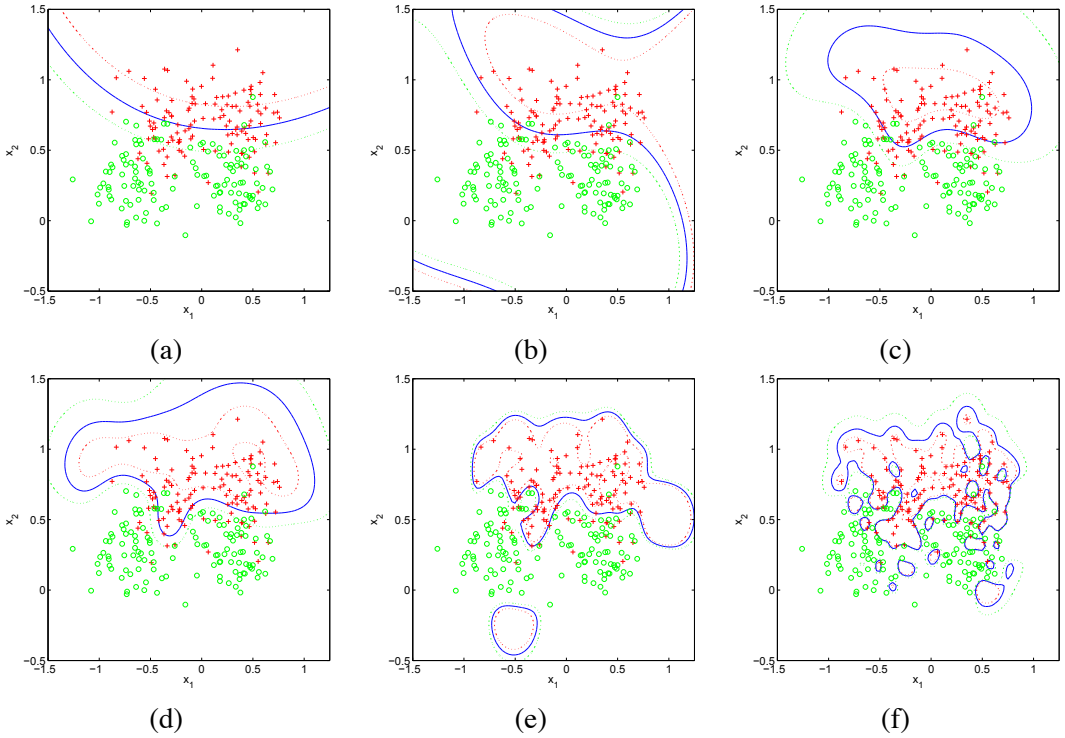
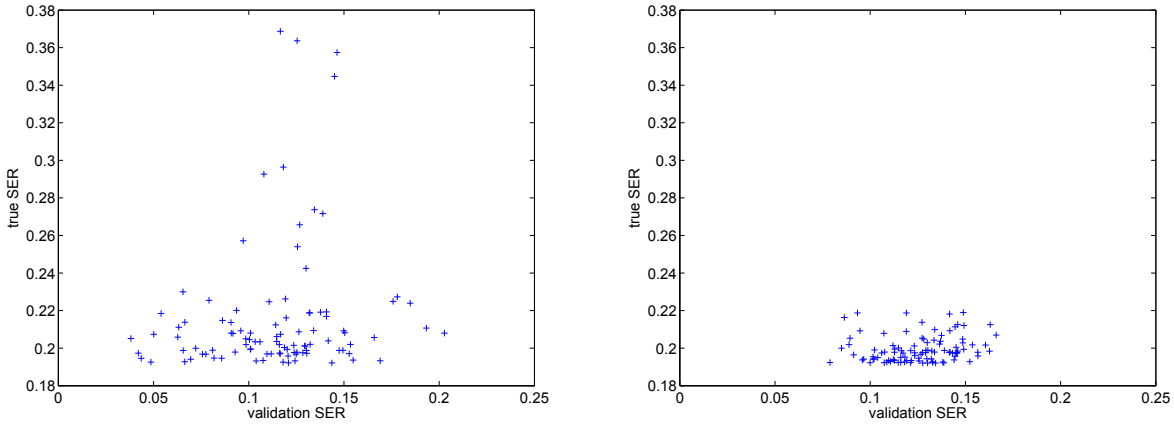


Figure 6: Kernel ridge regression models of the synthetic benchmark, using hyper-parameters selected according to the smoothed error rate over six random realisations of the validation set (shown in Figure 5). The variance of the model selection criterion can result in models ranging from under-fit, (a) and (b), through well-fitting, (c) and (d), to over-fit (e) and (f).

random realisations of the `banana` benchmark data set used in Rätsch et al. (2001) (see Section 5.1 for details). On 64 realisations of the data GPC out-performs KLR, but on 36 KLR out-performs GPC, even though the GPC is better on average (although the difference is not statistically significant in this case). If the classifiers had been evaluated on only one of the latter 36 realisations, it might incorrectly be concluded that the KLR classifier is superior to the GPC for that benchmark. However, it should also be noted that a difference in performance between two algorithms is unlikely to be of *practical* significance, even if it is *statistically* significant, if it is smaller than the variation in performance due to the random sampling of the data, as it is probable that a greater improvement in performance would be obtained by further data collection than by selection of the optimal classifier.

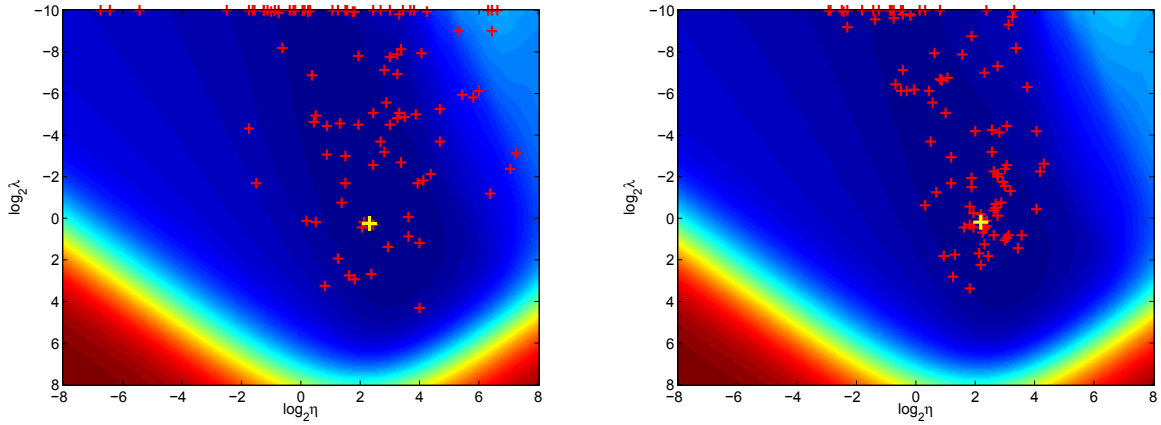
4.3 Is Over-fitting in Model Selection Really a Genuine Concern in Practice?

In the preceding part of this section we have demonstrated the deleterious effects of the variance of the model selection criterion using a synthetic benchmark data set, however this is not sufficient to establish that over-fitting in model selection is actually a genuine concern in practical applications or in the development of machine learning algorithms. Table 2 shows results obtained using



(a) (b)

Figure 7: Scatter plots of the true test smoothed error rate as a function of the validation set smoothed error rate for 100 randomly generated validation sets of (a) 64 and (b) 256 patterns.



(a) (b)

Figure 8: Contour plot of the mean validation set smoothed error rate over 100 randomly generated validation sets of (a) 64 and (b) 256 patterns. The minimum of the mean validation set error is marked by a large (yellow) cross, and the minimum for each realisation of the validation set marked by a small (red) cross.

kernel ridge regression (KRR) classifiers, with RBF and ARD kernel functions over the thirteen benchmarks described in Section 3.2. In each case, model selection was performed independently for each realisation of each benchmark by minimising the PRESS statistic using the Nelder-Mead simplex method (Nelder and Mead, 1965). For the majority of the benchmarks, a significantly lower

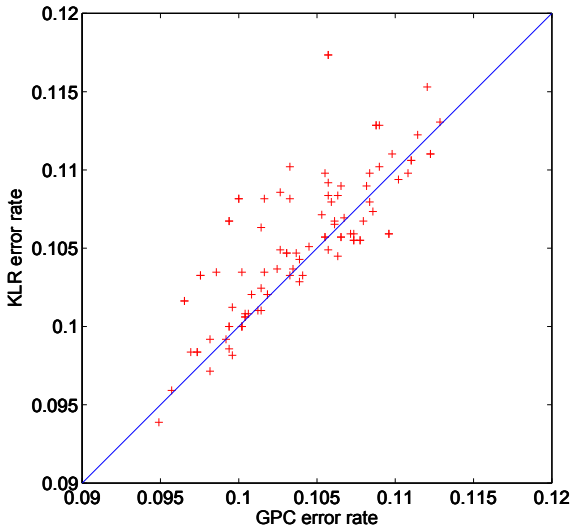


Figure 9: Scatter plots of the test set error for Gaussian process and Kernel Logistic regression classifiers (GPC and KLR respectively) for 100 realisations of the `banana` benchmark.

test error is achieved (according to the Wilcoxon signed ranks test) using the basic RBF kernel; the ARD kernel only achieves statistical superiority on one of the thirteen (`image`). This is perhaps a surprising result as the models are nested, the RBF kernel being a special case of the ARD kernel, so the optimal performance that can be achieved with the ARD kernel is guaranteed to be at least equal to the performance achievable using the RBF kernel. The reason for the poor performance of the ARD kernel in practice is because there are many more kernel parameters to be tuned in model selection and so many degrees of freedom available in optimising the model selection criterion. If the criterion used has a non-negligible variance, this includes optimisations exploiting the statistical peculiarities of the particular sample of data over which it is evaluated, and hence there will be more scope for over-fitting. Table 2 also shows the mean value of the PRESS statistic, following model selection, the fact that the majority of ARD models display a lower value for the PRESS statistic than the corresponding RBF model, while exhibiting a higher test error rate, is a strong indication of over-fitting the model selection criterion. This is a clear demonstration that over-fitting in model selection can be a significant problem in practical applications, especially where there are many hyper-parameters or where only a limited supply of data is available.

Table 3 shows the results of the same experiment performed using expectation-propagation based Gaussian process classifiers (EP-GPC) (Rasmussen and Williams, 2006), where the hyper-parameters are tuned independently for each realisation, for each benchmark individually by maximising the Bayesian evidence. While the leave-one-out cross-validation based PRESS criterion is known to exhibit a high variance, the variance of the evidence (which is also evaluated over a finite sample of data) is discussed less often. We find again here that the RBF covariance function often out-performs the more general ARD covariance function, and again the test error rate is often negatively correlated with the evidence for the models. This indicates that over-fitting the evidence is also a significant practical problem for the Gaussian process classifier.

Data Set	Test Error Rate		PRESS	
	RBF	ARD	RBF	ARD
banana	10.610 ± 0.051	10.638 ± 0.052	60.808 ± 0.636	60.957 ± 0.624
breast cancer	**26.727 ± 0.466**	28.766 ± 0.391	70.632 ± 0.328	**66.789 ± 0.385**
diabetis	**23.293 ± 0.169**	24.520 ± 0.215	146.143 ± 0.452	**141.465 ± 0.606**
flare solar	34.140 ± 0.175	34.375 ± 0.175	267.332 ± 0.480	**263.858 ± 0.550**
german	**23.540 ± 0.214**	25.847 ± 0.267	228.256 ± 0.666	**221.743 ± 0.822**
heart	**16.730 ± 0.359**	22.810 ± 0.411	42.576 ± 0.356	**37.023 ± 0.494**
image	2.990 ± 0.159	**2.188 ± 0.134**	74.056 ± 1.685	**44.488 ± 1.222**
ringnorm	**1.613 ± 0.015**	2.750 ± 0.042	28.324 ± 0.246	**27.680 ± 0.231**
splice	10.777 ± 0.144	9.943 ± 0.520	186.814 ± 2.174	**130.888 ± 6.574**
thyroid	4.747 ± 0.235	4.693 ± 0.202	9.099 ± 0.152	**6.816 ± 0.164**
titanic	22.483 ± 0.085	22.562 ± 0.109	48.332 ± 0.622	**47.801 ± 0.623**
twonorm	**2.846 ± 0.021**	4.292 ± 0.086	**32.539 ± 0.279**	35.620 ± 0.490
waveform	**9.792 ± 0.045**	11.836 ± 0.085	61.658 ± 0.596	**56.424 ± 0.637**

Table 2: Error rates of kernel ridge regression (KRR) classifier over thirteen benchmark data sets (Rätsch et al., 2001), using both standard radial basis function (RBF) and automatic relevance determination (ARD) kernels. Results shown in bold indicate an error rate that is statistically superior to that obtained with the same classifier using the other kernel function, or a PRESS statistic that is significantly lower.

Data Set	Test Error Rate		-Log Evidence	
	RBF	ARD	RBF	ARD
banana	**10.413 ± 0.046**	10.459 ± 0.049	116.894 ± 0.917	**116.459 ± 0.923**
breast cancer	**26.506 ± 0.487**	27.948 ± 0.492	110.628 ± 0.366	**107.181 ± 0.388**
diabetis	**23.280 ± 0.182**	23.853 ± 0.193	230.211 ± 0.553	**222.305 ± 0.581**
flare solar	34.200 ± 0.175	**33.578 ± 0.181**	394.697 ± 0.546	**384.374 ± 0.512**
german	**23.363 ± 0.211**	23.757 ± 0.217	359.181 ± 0.778	**346.048 ± 0.835**
heart	**16.670 ± 0.290**	19.770 ± 0.365	73.464 ± 0.493	**67.811 ± 0.571**
image	2.817 ± 0.121	**2.188 ± 0.076**	205.061 ± 1.687	**123.896 ± 1.184**
ringnorm	**4.406 ± 0.064**	8.589 ± 0.097	121.260 ± 0.499	**91.356 ± 0.583**
splice	11.609 ± 0.180	**8.618 ± 0.924**	365.208 ± 3.137	**242.464 ± 16.980**
thyroid	4.373 ± 0.219	4.227 ± 0.216	25.461 ± 0.182	**18.867 ± 0.170**
titanic	22.637 ± 0.134	22.725 ± 0.133	78.952 ± 0.670	**78.373 ± 0.683**
twonorm	**3.060 ± 0.034**	4.025 ± 0.068	45.901 ± 0.577	**42.044 ± 0.610**
waveform	**10.100 ± 0.047**	11.418 ± 0.091	105.925 ± 0.954	**91.239 ± 0.962**

Table 3: Error rates of expectation propagation based Gaussian process classifiers (EP-GPC), using both standard radial basis function (RBF) and automatic relevance determination (ARD) kernels. Results shown in bold indicate an error rate that is statistically superior to that obtained with the same classifier using the other kernel function or evidence that is significantly higher.

4.4 Avoiding Over-fitting in Model Selection

It seems reasonable to suggest that over-fitting in model selection is possible whenever a model selection criterion evaluated over a finite sample of data is directly optimised. Like over-fitting in training, over-fitting in model selection is likely to be most severe when the sample of data is small and the number of hyper-parameters to be tuned is relatively large. Likewise, assuming additional data are unavailable, potential solutions to the problem of over-fitting the model selection criterion are likely to be similar to the tried and tested solutions to the problem of over-fitting the training criterion, namely regularisation (Cawley and Talbot, 2007), early stopping (Qi et al., 2004) and model or hyper-parameter averaging (Cawley, 2006; Hall and Robinson, 2009). Alternatively, one might minimise the number of hyper-parameters, for instance by treating kernel parameters as simply parameters and optimising them at the first level of inference and have a single regularisation hyper-parameter controlling the overall complexity of the model. For very small data sets, where the problem of over-fitting in both learning and model selection is greatest, the preferred approach would be to eliminate model selection altogether and opt for a fully Bayesian approach, where the hyper-parameters are integrated out rather than optimised (e.g., Williams and Barber, 1998). Another approach is simply to avoid model selection altogether using an ensemble approach, for example the Random Forest (RF) method (Breiman, 2001). However, while such methods often achieve state-of-the-art performance, it is often easier to build expert knowledge into hierarchical models, for example through the design of kernel or covariance functions, so unfortunately approaches such as the RF are not a panacea.

While the problem of over-fitting in model selection is of the same nature as that of over-fitting at the first level of inference, the lack of mathematical tractability appears to have limited the theoretical analysis of model selection via optimisation of a model selection criterion. For example, regarding leave-one-out cross-validation, Kulkarni et al. (1998) comment "In spite of the practical importance of this estimate, relatively little is known about its properties. *The available theory is especially poor when it comes to analysing parameter selection based on minimizing the deleted estimate*." (our emphasis). While some asymptotic results are available (Stone, 1977; Shao, 1993; Toussaint, 1974), these are not directly relevant to the situation considered here, where over-fitting occurs due to optimising the values of hyper-parameters using a model selection criterion evaluated over a finite, often quite limited, sample of data. Estimates of the variance of the cross-validation error are available for some models (Luntz and Brailovsky, 1969; Vapnik, 1982), however Bengio and Grandvalet (2004) have shown there is no unbiased estimate of the variance of (k-fold) cross-validation. More recently bounds on the error of leave-one-out cross-validation based on the idea of *stability* have been proposed (Kearns and Ron, 1999; Bousquet and Elisseeff, 2002; Zhang, 2003). In this section, we have demonstrated that over-fitting in model selection is a genuine problem in machine learning, and hence is likely to be an area that could greatly benefit from further theoretical analysis.

5. Bias in Performance Estimation

Avoiding potentially significant bias in performance evaluation, arising due to over-fitting in model selection, is conceptually straightforward. The key is to treat both training *and* model selection together, as integral parts of the model fitting procedure and ensure they are never performed separately at any point of the evaluation process. We present two examples of potentially biased evaluation protocols that do not adhere to this principle. The scale of the bias observed on some data sets

is much larger than the difference in performance between learning algorithms, and so one could easily draw incorrect inferences based on the results obtained. This highlights the importance of this issue in empirical studies. We also demonstrate that the magnitude of the bias depends on the learning and model selection algorithms involved in the comparison and that combinations that are more prone to over-fitting in model selection are favored by biased protocols. This means that studies based on potentially biased protocols are not internally consistent, even if it is acknowledged that a bias with respect to other studies may exist.

5.1 An Unbiased Performance Evaluation Methodology

We begin by describing an unbiased performance protocol, that correctly accounts for any over-fitting that may occur in model selection. Three classifiers are evaluated using an unbiased protocol, in which model selection is performed separately for each realisation of each data set. This is termed the "internal" protocol as the model selection process is performed independently within each fold of the resampling procedure. In this way, the performance estimate includes a component properly accounting for the error introduced by over-fitting the model selection criterion. The classifiers used were as follows: RBF-KRR—kernel ridge regression with a radial basis function kernel, with model selection based on minimisation of Allen's PRESS statistic, as described in Section 2. RBF-KLR—kernel logistic regression with a radial basis function kernel and model selection based on an approximate leave-one-out cross-validation estimate of the log-likelihood (Cawley and Talbot, 2008). EP-GPC—expectation-propagation based Gaussian process classifier, with an isotropic squared exponential covariance function, with model selection based on maximising the marginal likelihood (e.g., Rasmussen and Williams, 2006). The mean error rates obtained using these classifiers under an unbiased protocol are shown in Table 4. In this case, the mean ranks of all methods are only minimally different, and so there is little if any evidence for a statistically significant superiority of any of the classifiers over any other. Figure 10 shows a critical difference diagram (Demšar, 2006), providing a graphical illustration of this result. A critical difference diagram displays the mean rank of a set of classifiers over a suite of benchmark data sets, with cliques of classifiers with statistically similar performance connected by a bar. The critical difference in average ranks required for a statistical superiority of one classifier over another is also shown, labelled "CD".

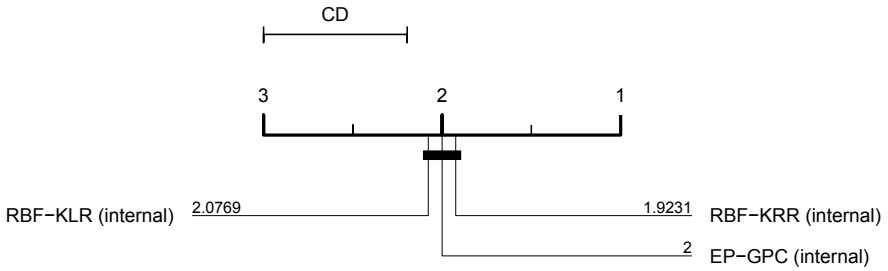


Figure 10: Critical difference diagram (Demšar, 2006) showing the average ranks of three classifiers with internal model selection protocol.

Data Set	GPC (internal)	KLR (internal)	KRR (internal)
banana	10.413 ± 0.046	10.567 ± 0.051	10.610 ± 0.051
breast cancer	26.506 ± 0.487	26.636 ± 0.467	26.727 ± 0.466
diabetis	23.280 ± 0.182	23.387 ± 0.180	23.293 ± 0.169
flare solar	34.200 ± 0.175	34.197 ± 0.170	34.140 ± 0.175
german	23.363 ± 0.211	23.493 ± 0.208	23.540 ± 0.214
heart	16.670 ± 0.290	16.810 ± 0.315	16.730 ± 0.359
image	2.817 ± 0.121	3.094 ± 0.130	2.990 ± 0.159
ringnorm	4.406 ± 0.064	1.681 ± 0.031	1.613 ± 0.015
splice	11.609 ± 0.180	11.248 ± 0.177	10.777 ± 0.144
thyroid	4.373 ± 0.219	4.293 ± 0.222	4.747 ± 0.235
titanic	22.637 ± 0.134	22.473 ± 0.103	22.483 ± 0.085
twonorm	3.060 ± 0.034	2.944 ± 0.042	2.846 ± 0.021
waveform	10.100 ± 0.047	9.918 ± 0.043	9.792 ± 0.045

Table 4: Error rate estimates of three classifiers over a suite of thirteen benchmark data sets: The results for each method are presented in the form of the mean error rate over test data for 100 realisations of each data set (20 in the case of the `image` and `splice` data sets), along with the associated standard error.

It is not unduly surprising that there should be little evidence for any statistically significant superiority, as all three methods give rise to structurally similar models. The models though differ significantly in their model selection procedures, the EP-GPC is based on stronger statistical assumptions, and so can be expected to excel where these assumptions are justified, but poorly where the model is mis-specified (e.g., the ringnorm benchmark). The cross-validation based model selection procedures, on the other hand, are more pragmatic and being based on much weaker assumptions might be expected to provide a more consistent level of accuracy.

5.2 An Example of Biased Evaluation Methodology

The performance evaluation protocol most often used in conjunction with the suite of benchmark data sets, described in Section 3.2, seeks to perform model selection independently for only the first five realisations of each data set. The median values of the hyper-parameters over these five folds are then determined and subsequently used to evaluate the error rates for each realisation. This "median" performance evaluation protocol was introduced in the same paper that popularised this suite of benchmark data sets (Rätsch et al., 2001) and has been widely adopted (e.g., Mika et al., 1999; Weston, 1999; Billings and Lee, 2002; Chapelle et al., 2002; Chu et al., 2003; Stewart, 2003; Mika et al., 2003; Gold et al., 2005; Peña Centeno and D., 2006; Andelić et al., 2006; An et al., 2007; Chen et al., 2009). The original motivation for this protocol was that the internal model selection protocol was prohibitively expensive using workstations available (Rätsch et al., 2001), which was perfectly reasonable at the time, but is no longer true.[3] The use of the median, however, can be expected to introduce an optimistic bias into the performance estimates obtained using this "median" protocol. Firstly all of the training data comprising the first five realisations have been

3. All of the experimental results presented in this paper were obtained using a single modern Linux workstation.

Data Set	KRR (internal)	KRR (median)	Bias
banana	10.610 ± 0.051	10.384 ± 0.042	0.226 ± 0.034
breast cancer	26.727 ± 0.466	26.377 ± 0.441	0.351 ± 0.195
diabetis	23.293 ± 0.169	23.150 ± 0.157	0.143 ± 0.074
flare solar	34.140 ± 0.175	34.013 ± 0.166	0.128 ± 0.082
german	23.540 ± 0.214	23.380 ± 0.220	0.160 ± 0.067
heart	16.730 ± 0.359	15.720 ± 0.306	1.010 ± 0.186
image	2.990 ± 0.159	2.802 ± 0.129	0.188 ± 0.095
ringnorm	1.613 ± 0.015	1.573 ± 0.010	0.040 ± 0.010
splice	10.777 ± 0.144	10.763 ± 0.137	0.014 ± 0.055
thyroid	4.747 ± 0.235	4.560 ± 0.200	0.187 ± 0.100
titanic	22.483 ± 0.085	22.407 ± 0.102	0.076 ± 0.077
twonorm	2.846 ± 0.021	2.868 ± 0.017	-0.022 ± 0.014
waveform	9.792 ± 0.045	9.821 ± 0.039	-0.029 ± 0.020

Table 5: Error rate estimates of three classifiers over a suite of thirteen benchmark data sets: The results for each method are presented in the form of the mean error rate over test data for 100 realisations of each data set (20 in the case of the image and splice data sets), along with the associated standard error.

used during the model selection process for the classifiers used in every fold of the re-sampling. This means that some of the test data for each fold is no longer statistically "pure" as it has been seen during model selection. Secondly, and more importantly, the median operation acts as a variance reduction step, so the median of the five sets of hyper-parameters is likely to be better on average than any of the five from which it is derived. Lastly, as the hyper-parameters are now fixed, there is no longer scope for over-fitting the model selection criterion due to peculiarities of the sampling of data for the training and test partitions in each realisation.

We begin by demonstrating that the results using the internal and median protocols are not commensurate, and so the results obtained using different methods are not directly comparable. Table 5 shows the error rate obtained using the RBF-KRR classifier with the internal and median performance evaluation protocols and the resulting bias, that is, the difference between the mean error rates obtained with the internal and median protocols. It is clearly seen that the median protocol introduces a positive bias on almost all benchmarks (`twonorm` and `waveform` being the exceptions) and that the bias can be quite substantial on some benchmarks. Indeed, for several benchmarks, `breast cancer`, `german`, `heart` and `thyroid` in particular, the bias is larger than the typical difference in performance between classifiers evaluated using an unbiased protocol. Demšar (2006) recommends the Wilcoxon signed ranks test for determination of the statistical significance of the superiority of one classifier over another over multiple data sets. Applying this test to the data shown for EP-GPC (internal), RBF-KLR (internal) and RBF-KRR (median), from Tables 4 and 5, reveals that the RBF-KRR (median) classifier is statistically superior to the remaining classifiers, at the 95% level of significance. A critical difference diagram summarising this result is shown in Figure 12. However, the difference in performance is entirely spurious as it is purely the result of reducing the effects of over-fitting in model selection and does not reflect the true operational performance of the combination of classifier and model selection method. It is clear then that results obtained using

the internal and median protocols are not directly comparable, and so reliable inferences cannot be drawn by comparison of results from different studies, using biased and unbiased protocols.

5.2.1 IS THE BIAS SOLELY DUE TO INADVERTENT RE-USE OF TEST SAMPLES?

One explanation for the observed bias of the median protocol is that some of the training samples for the first five realisations of the benchmark, which have been used in tuning the hyper-parameters, also appear in the test sets for other realisations of the benchmark used for performance analysis. In this section, we demonstrate that this inadvertent re-use of test samples is not the only cause of the bias. One hundred replications of the internal and median protocol were performed using the `synthetic` benchmark, for which an inexhaustible supply of i.i.d. data is available. However, in this case in each realisation, 100 training sets of 64 patterns and a large test set of 4096 samples were generated, all mutually disjoint. This means the only remaining source of bias is the amelioration of over-fitting in model selection by the reduction of variance by taking the median of the hyper-parameters over the first five folds (cf. Hall and Robinson, 2009). Figure 11 shows the mean test errors for the internal and median protocols over 100 replications, showing a very distinct optimistic bias in the median protocol (statistically highly significant according to the Wilcoxon signed ranks test, $p < 0.001$), even though there is absolutely no inadvertent re-use of test data.

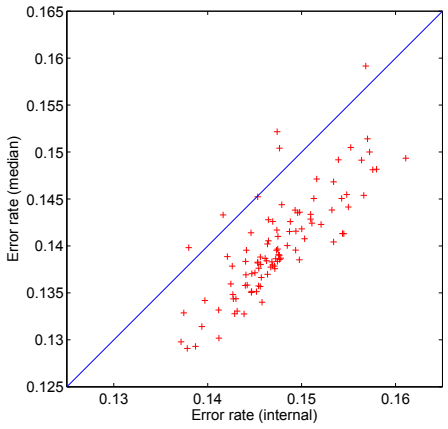


Figure 11: Mean error rates for the internal and median evaluation protocols for the `synthetic` benchmark, without inadvertent re-use of test data.

5.2.2 IS THE MEDIAN PROTOCOL INTERNALLY CONSISTENT?

Having established that the median protocol introduces an optimistic bias, and that the results obtained using the internal and median protocols do not give comparable results, we next turn our attention to whether the median protocol is internally consistent, that is, does the median protocol give the correct rank order of the classifiers? Table 6 shows the performance of three classifiers evaluated using the median protocol; the corresponding critical difference diagram is shown in Figure 13. In this case the difference in performance between classifiers is not statistically significant according to the Friedman test, however it can clearly be seen that the bias of the median protocol

Data Set	EP-GPC (median)	RBF-KLR (median)	RBF-KRR (median)
banana	10.371 ± 0.045	10.407 ± 0.047	10.384 ± 0.042
breast cancer	26.117 ± 0.472	26.130 ± 0.474	26.377 ± 0.441
diabetis	23.333 ± 0.191	23.300 ± 0.177	23.150 ± 0.157
flare solar	34.150 ± 0.170	34.212 ± 0.176	34.013 ± 0.166
german	23.160 ± 0.216	23.203 ± 0.218	23.380 ± 0.220
heart	16.400 ± 0.273	16.120 ± 0.295	15.720 ± 0.306
image	2.851 ± 0.102	3.030 ± 0.120	2.802 ± 0.129
ringnorm	4.400 ± 0.064	1.574 ± 0.011	1.573 ± 0.010
splice	11.607 ± 0.184	11.172 ± 0.168	10.763 ± 0.137
thyroid	4.307 ± 0.217	4.040 ± 0.221	4.560 ± 0.200
titanic	22.490 ± 0.095	22.591 ± 0.135	22.407 ± 0.102
twonorm	3.241 ± 0.039	3.068 ± 0.033	2.868 ± 0.017
waveform	10.163 ± 0.045	9.888 ± 0.042	9.821 ± 0.039

Table 6: Error rate estimates of three classifiers over a suite of thirteen benchmark data sets: The results for each method are presented in the form of the mean error rate over test data for 100 realisations of each data set (20 in the case of the image and splice data sets), along with the associated standard error.

has favored one classifier, namely the RBF-KRR, much more strongly than the others. It seems feasible then that the bias of the median protocol may be sufficient in other cases to amplify a small difference in performance, due perhaps to an accidentally favorable choice of data sets, to the point where it spuriously appears to be statistically significant. This suggests that the median protocol may be unreliable and perhaps should be deprecated.

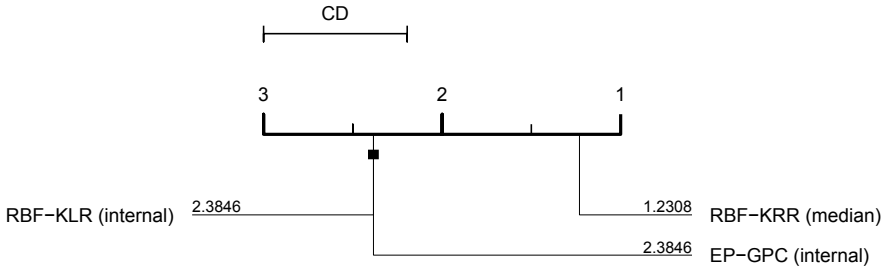


Figure 12: Critical difference diagram (Demšar, 2006) showing the average ranks of three classifiers, EP-GPC and RBF-KLR with internal model selection protocol and RBF-KLR using the optimistically biased median protocol (cf. Figure 10).

Next, we perform a statistical analysis to determine whether there is a statistically significant difference in the magnitude of the biases introduced by the median protocol for different classifiers,

Data Set	RBF-KRR bias	RBF-EP-GPC bias	Wilcoxon p-value
banana	0.226 ± 0.034	0.043 ± 0.012	< 0.05
breast cancer	0.351 ± 0.195	0.390 ± 0.186	0.934
diabetis	0.143 ± 0.074	-0.053 ± 0.051	< 0.05
flare solar	0.128 ± 0.082	0.050 ± 0.090	0.214
german	0.160 ± 0.067	0.203 ± 0.051	0.458
heart	1.010 ± 0.186	0.270 ± 0.120	< 0.05
image	0.188 ± 0.095	-0.035 ± 0.032	0.060
ringnorm	0.040 ± 0.010	0.006 ± 0.002	< 0.05
splice	0.014 ± 0.055	0.002 ± 0.014	0.860
thyroid	0.187 ± 0.100	0.067 ± 0.064	0.159
titanic	0.076 ± 0.077	0.147 ± 0.090	0.846
twonorm	-0.022 ± 0.014	-0.180 ± 0.032	< 0.05
waveform	-0.029 ± 0.020	-0.064 ± 0.022	0.244

Table 7: Results of a statistical analysis of the bias introduced by the median protocol into the test error rates for RBF-KRR and RBF-EP-GPC, using the Wilcoxon signed ranks test.

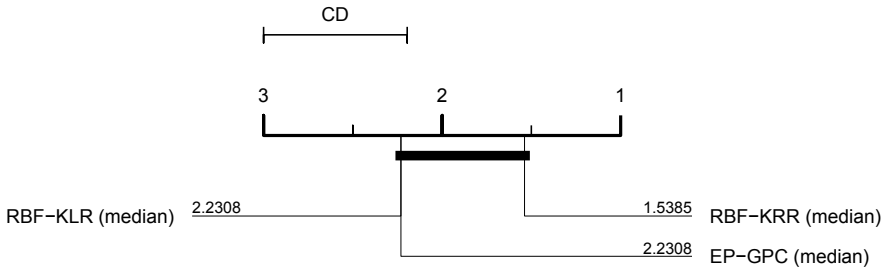


Figure 13: Critical difference diagram showing the average ranks of three classifiers with the median model selection protocol (cf. Figure 10).

for each benchmark data set.[4] First the bias introduced by the use of the median protocol was computed for the RBF KRR and RBF EP-GPC classifiers as the difference between the test set error estimated by the internal and median protocols. The Wilcoxon signed rank test was then used to determine whether there is a statistically significant difference in the bias, over the 100 realisations of the benchmark (20 in the case of the `image` and `splice` benchmarks) . The results obtained are shown in Table 7, the p-value is below 0.05 for five of the thirteen benchmarks, indicating that in each case the median protocol is significantly biased in favour of the RBF KRR classifier. Clearly, as the median protocol does not impose a commensurate bias on the estimated test error rates for different classifiers, it does not provide a reliable protocol for comparing the performance of machine learning algorithms.

4. We are grateful to an anonymous reviewer for suggesting this particular form of analysis.

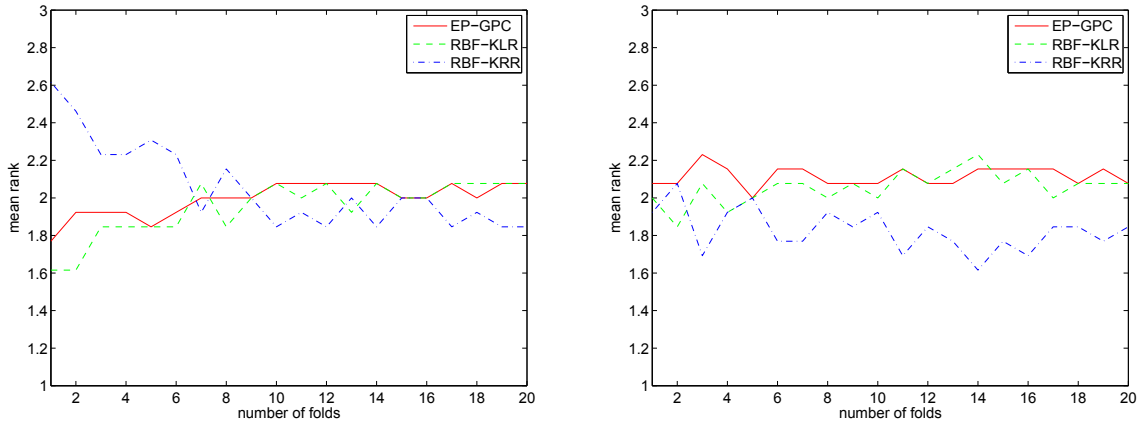


Figure 14: Mean ranks of three classifiers as a function of the number of folds used in the repeated split sample model selection procedure employed by the kernel ridge regression (RBF-KRR) machine, using (a) the unbiased *internal* protocol and (b) the biased *median* protocol.

In the final illustration of this section, we show that the magnitude of the bias introduced by the median protocol is greater for model selection criteria with a high variance. This means the median protocol favors most the least reliable model selection procedures and as a result does not provide a reliable indicator even of relative performance of classifier-model selection procedures combinations. Again the RBF-KRR model is used as the base classifier, however in this case a repeated split-sample model selection criterion is used, where the data are repeatedly split at random to form disjoint training and validation sets in proportions 9:1, and the hyper-parameters tuned to optimise the average mean-squared error over the validation sets. In this way, the variance of the model selection criterion can be controlled by varying the number of repetitions, with the variance decreasing as the number of folds becomes larger. Figure 14 (a) shows a plot of the average ranks of EP-GPC and RBF-KLR classifiers, with model selection performed as in previous experiments, and RBF-KRR with repeated split-sample model selection, as a function of the number of folds. In each case the unbiased internal evaluation protocol was used. Clearly if the number of folds is small (five or less), the RBF-KRR model performs poorly, due to over-fitting in model selection due to the high variance of the criterion used. However, as the number of folds increases, the variance of the model selection criterion falls, and the performances of all three algorithms are very similar. Figure 14 (b) shows the corresponding result using the biased median protocol. The averaging of hyper-parameters reduces the apparent variance of the model selection criterion, and this disguises the poor performance of the RBF-KRR model when the number of folds is small. This demonstrates that the bias introduced by the median protocol favors most the worst model selection criterion, which is a cause for some concern.

Data Set	External	Internal	Bias
banana	10.355 ± 0.146	10.495 ± 0.158	0.140 ± 0.035
breast cancer	26.280 ± 0.232	27.470 ± 0.250	1.190 ± 0.135
diabetis	22.891 ± 0.127	23.056 ± 0.134	0.165 ± 0.050
flare solar	34.518 ± 0.172	34.707 ± 0.179	0.189 ± 0.051
german	23.999 ± 0.117	24.217 ± 0.125	0.219 ± 0.045
heart	16.335 ± 0.214	16.571 ± 0.220	0.235 ± 0.073
image	3.081 ± 0.102	3.173 ± 0.112	0.092 ± 0.035
ringnorm	1.567 ± 0.058	1.607 ± 0.057	0.040 ± 0.014
splice	10.930 ± 0.219	11.170 ± 0.280	0.240 ± 0.152
thyroid	3.743 ± 0.137	4.279 ± 0.152	0.536 ± 0.073
titanic	22.167 ± 0.434	22.487 ± 0.442	0.320 ± 0.077
twonorm	2.480 ± 0.067	2.502 ± 0.070	0.022 ± 0.021
waveform	9.613 ± 0.168	9.815 ± 0.183	0.203 ± 0.064

Table 8: Error rate estimates for kernel ridge regression over thirteen benchmark data sets, for model selection schemes that are internal and external to the cross-validation process. The results for each approach and the relative bias are presented in the form of the mean error rate over for 100 realisations of each data set (20 in the case of the image and splice data sets), along with the associated standard error.

5.3 Another Example of Biased Evaluation Methodology

In a biased evaluation protocol, occasionally observed in machine learning studies, an initial model selection step is performed using all of the available data, often interactively as part of a "preliminary study". The data are then repeatedly re-partitioned to form one or more pairs of random, disjoint design and test sets. These are then used for performance evaluation *using the same fixed set of hyper-parameter values*. This practice may seem at first glance to be fairly innocuous, however the test data are no longer statistically pure, as they have been "seen" by the models in tuning the hyper-parameters. This would not present a serious problem were it not for the danger of over-fitting in model selection, which means that in practice the hyper-parameters will inevitably be tuned to an extent in ways that take advantage of the statistical peculiarities of this particular set of data rather than only in ways that favor improved generalisation. As a result the hyper-parameter settings retain a partial "memory" of the data that now form the test partition. We should therefore expect to observe an optimistic bias in the performance estimates obtained in this manner.

Table 8 shows a comparison of 10-fold cross-validation estimates of the test error rate, for kernel ridge regression with a Gaussian radian basis function kernel, obtained using protocols where the model selection stage is either *external* or *internal* to the cross-validation procedure. In the external protocol, model selection is performed once using the entire design set, as described above. In the internal protocol, the model selection step is performed separately in each fold of the cross-validation. The internal cross-validation procedure therefore provides a more realistic estimate of the performance of the combination of model selection and learning algorithm that is actually used to construct the final model. The table also shows the relative bias (i.e., the mean difference between the internal and external cross-validation protocols). The external protocol clearly exhibits a consistently optimistic bias with respect to the more rigorous internal cross-validation protocol, over

all thirteen benchmarks. Furthermore, the bias is statistically significant (i.e., larger than twice the standard error of the estimate) for all benchmarks, apart from `splice` and `twonorm`. In many cases, the bias is of similar magnitude to the typical difference observed between competitive learning algorithms (cf. Table 4). In some cases, for example, `banana` and `thyroid` benchmarks, the bias is of a surprising magnitude, likely to be large enough to conceal even the true difference between even state-of-the-art and uncompetitive learning algorithms. This clearly shows that the external cross-validation protocol exhibits a consistent optimistic bias, potentially of a very substantial magnitude even when the number of hyper-parameters is small (in this case only two), and so should not be used in practice.

6. Conclusions

In this paper, we have discussed the importance of bias and variance in model selection and performance evaluation, and demonstrated that a high variance can lead to over-fitting in model selection, and hence poor performance, even when the number of hyper-parameters is relatively small. Furthermore, we have shown that a potentially severe form of selection bias can be introduced into performance evaluation by protocols that have been adopted in a number of existing empirical studies. Fortunately, it seems likely that over-fitting in model selection can be overcome using methods that have already been effective in preventing over-fitting during training, such as regularisation or early stopping. Little attention has so far been focused on over-fitting in model selection, however in this paper we have shown that it presents a genuine pitfall in the practical application of machine learning algorithms and in empirical comparisons. In order to overcome the bias in performance evaluation, model selection should be viewed as an integral part of the model fitting procedure, and should be conducted independently in each trial in order to prevent selection bias and because it reflects best practice in operational use. Rigorous performance evaluation therefore requires a substantial investment of processor time in order to evaluate performance over a wide range of data sets, using multiple randomised partitionings of the available data, with model selection performed separately in each trial. However, it is straightforward to fully automate these steps, and so requires little manual involvement. Performance evaluation according to these principles requires repeated training of models using different sets of hyper-parameter values on different samples of the available data, and so is also well-suited to parallel implementation. Given the recent trend in processor design towards multi-core designs, rather than faster processor speeds, rigorous performance evaluation is likely to become less and less time-consuming, and so there is little justification for the continued use of potentially biased protocols.

Acknowledgments

The authors would like to thank Gareth Janacek, Wenjia Wang and the anonymous reviewers for their helpful comments on earlier drafts of this paper, and the organisers and participants of the WCCI-2006 Performance Prediction Challenge and workshop that provided the inspiration for our work on model selection and performance prediction. G. C. Cawley is supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/F010508/1 - Advancing Machine Learning Methodology for New Classes of Prediction Problems.

References

D. M. Allen. The relationship between variable selection and prediction. *Technometrics*, 16:125–127, 1974.

C. Ambroise and G. J. McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences*, 99(10):6562–6566, May 14 2002. doi: 10.1073/pnas.102102699.

S. An, W. Liu, and S. Venkatesh. Fast cross-validation algorithms for least squares support vector machines and kernel ridge regression. *Pattern Recognition*, 40(8):2154–2162, August 2007. doi: 10.1016/j.patcog.2006.12.015.

E. Andelić, M. Schafföner, M. Katz, S. E. Krüger, and A. Wendermuth. Kernel least-squares models using updates of the pseudoinverse. *Neural Computation*, 18(12):2928–2935, December 2006. doi: 10.1162/neco.2006.18.12.2928.

Y. Bengio and Y. Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105, 2004.

S. A. Billings and K. L. Lee. Nonlinear Fisher discriminant analysis using a minimum squared error cost function and the orthogonal least squares algorithm. *Neural Networks*, 15(2):263–270, March 2002. doi: 10.1016/S0893-6080(01)00142-3.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

L. Bo, L. Wang, and L. Jiao. Feature scaling for kernel Fisher discriminant analysis using leave-one-out cross validation. *Neural Computation*, 18(4):961–978, April 2006. doi: 10.1162/neco.2006.18.4.961.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001. doi: 10.1023/A:1010933404324.

G. C. Cawley. Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs. In *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks (IJCNN-06)*, pages 1661–1668, Vancouver, BC, Canada, July 16–21 2006. doi: 10.1109/IJCNN.2006.246634.

G. C. Cawley and N. L. C. Talbot. Efficient leave-one-out cross-validation of kernel Fisher discriminant classifiers. *Pattern Recognition*, 36(11):2585–2592, November 2003. doi: 10.1016/S0031-3203(03)00136-5.

G. C. Cawley and N. L. C. Talbot. Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, April 2007.

G. C. Cawley and N. L. C. Talbot. Efficient approximate leave-one-out cross-validation for kernel logistic regression. *Machine Learning*, 71(2–3):243–264, June 2008. doi: 10.1007/s10994-008-5055-9.

G. C. Cawley, G. J. Janacek, and N. L. C. Talbot. Generalised kernel machines. In *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks (IJCNN-07)*, pages 1720–1725, Orlando, Florida, USA, August 12–17 2007. doi: 10.1109/IJCNN.2007.4371217.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, January 2002. doi: 10.1023/A:1012450327387.

H. Chen, P. Tino, and X. Yao. Probabilistic classification vector machines. *IEEE Transactions on Neural Networks*, 20(6):901–914, June 2009. doi: 10.1109/TNN.2009.2014161.

W. Chu, S. S. Keerthi, and C. J. Ong. Bayesian trigonometric support vector classifier. *Neural Computation*, 15(9):2227–2254, September 2003. doi: 10.1162/089976603322297368.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, second edition, 2001.

B. Efron and R. J. Tibshirani. *Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability. Chapman & Hall, 1994.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, January 1992. doi: 10.1162/neco.1992.4.1.1.

C. Gold, A. Holub, and P. Sollich. Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. *Neural Networks*, 18(5):693–701, July/August 2005. doi: 10.1016/j.neunet.2005.06.044.

I. Guyon, A. Saffari, G. Dror, and G. Cawley. Model selection: Beyond the Bayesian/frequentist divide. *Journal of Machine Learning Research*, 11:61–87, 2009.

P. Hall and A. P. Robinson. Reducing the variability of crossvalidation for smoothing parameter choice. *Biometrika*, 96(1):175–186, March 2009. doi: doi:10.1093/biomet/asn068.

M. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11(6):1427–1453, August 1999. doi: 10.1162/089976699300016304.

G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

S. R. Kulkarni, G. Lugosi, and S. S. Venkatesh. Learning pattern classification — a survey. *IEEE Transactions on Information Theory*, 44(6):2178–2206, October 1998.

P. A. Lachenbruch and M. R. Mickey. Estimation of error rates in discriminant analysis. *Technometrics*, 10(1):1–12, February 1968.

A. Luntz and V. Brailovsky. On estimation of characters obtained in statistical procedure of recognition (in Russian). *Techicheskaya Kibernetica*, 3, 1969.

D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, May 1992. doi: 10.1162/neco.1992.4.3.415.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London, Series A*, 209:415–446, 1909.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48, Maddison, WI, USA, 21–25 August 1999. doi: 10.1109/NNSP.1999.788121.

S. Mika, G. Rätsch, J. Weston, B. Schölkpf, and K.-R. Müller. Contructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):623–628, May 2003. doi: 10.1109/TPAMI.2003.1195996.

J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7: 308–313, 1965.

T. Peña Centeno and Lawrence N. D. Optimising kernel parameters and regularisation coefficients for non-linear discriminant analysis. *Journal of Machine Learning Research*, 7:455–491, February 2006.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9): 1481–1497, September 1990. doi: 10.1109/5.58326.

Y. Qi, T. P. Minka, R. W. Picard, and Z. Ghahramani. Predictive automatic relevance determination by expectation propagation. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 671–678, Banff, Alberta, Canada, July 4–8 2004.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3): 287–320, March 2001. doi: 10.1023/A:1007618119488.

R. M. Rifkin and R. A. Lippert. Notes on regularized least squares. Technical Report MIT-CSAIL-TR-2007-025, Computer Science and Artificial Intelligence Laboratory, MIT, May 2007.

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 515–521. Morgan Kaufmann, 1998.

J. Shao. Linear model selection by cross-validation. *Journal of the American Statistical Society*, 88:486–494, 1993.

I. Stewart. On the optimal parameter choice for ν-support vector machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1274–1284, October 2003. doi: 10.1109/TPAMI.2003.1233901.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 36(2):111–147, 1974.

M. Stone. Asymptotics for and against cross-validation. *Biometrika*, 64(1):29–35, April 1977. doi: 10.1093/biomet/64.1.29.

J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vanderwalle. *Least Squares Support Vector Machine*. World Scientific Publishing Company, Singapore, 2002. ISBN 981-238-151-1.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. John Wiley, New York, 1977.

G. Toussaint. Bibliography on estimation of misclassification. *IEEE Transactions on Information Theory*, IT-20(4):472–479, July 1974.

V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, 1982.

V. N. Vapnik. *Statistical Learning Theory*. Adaptive and learning systems for signal processing, communications and control series. Wiley, 1998.

S. Weisberg. *Applied Linear Regression*. Probability and Mathematical Statistics. John Wiley & Sons, second edition, 1985.

J. Weston. Leave-one-out support vector machines. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 727–733, San Fransisco, CA, USA, 1999. Morgan Kaufmann.

C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, December 1998. doi: 10.1109/34.735807.

P. M. Williams. A Marquardt algorithm for choosing the step size in backpropagation learning with conjugate gradients. Technical Report CSRP-229, University of Sussex, February 1991.

T. Zhang. Leave-one-out bounds for kernel machines. *Neural Computation*, 15(6):1397–1437, June 2003. doi: 10.1162/089976603321780326.

Journal of Machine Learning Research 11 (2010) 2109-2113 Submitted 2/10; Revised 4/10; Published 8/10

Model-based Boosting 2.0

Torsten Hothorn TORSTEN.HOTHORN@R-PROJECT.ORG
Institut für Statistik
LMU München
DE-80539 München

Peter Bühlmann BUHLMANN@STAT.MATH.ETHZ.CH
Seminar für Statistik
ETH Zürich
CH-8092 Zürich

Thomas Kneib THOMAS.KNEIB@UNI-OLDENBURG.DE
Institut für Mathematik
Universität Oldenburg
DE-26111 Oldenburg

Matthias Schmid MATTHIAS.SCHMID@IMBE.MED.UNI-ERLANGEN.DE
Benjamin Hofner BENJAMIN.HOFNER@IMBE.MED.UNI-ERLANGEN.DE
Institut für Medizininformatik, Biometrie und Epidemiologie
FAU Erlangen-Nürnberg
DE-91054 Erlangen

Editor: Mikio Braun

Abstract

We describe version 2.0 of the R add-on package *mboost*. The package implements boosting for optimizing general risk functions using component-wise (penalized) least squares estimates or regression trees as base-learners for fitting generalized linear, additive and interaction models to potentially high-dimensional data.

Keywords: component-wise functional gradient descent, splines, decision trees

1. Overview

The R add-on package *mboost* (Hothorn et al., 2010) implements tools for fitting and evaluating a variety of regression and classification models that have been suggested in machine learning and statistics. Optimization within the empirical risk minimization framework is performed via a component-wise functional gradient descent algorithm. The algorithm originates from the statistical view on boosting algorithms (Friedman et al., 2000; Bühlmann and Yu, 2003). The theory and its implementation in *mboost* allow for fitting complex prediction models, taking potentially many interactions of features into account, as well as for fitting additive and linear models. The model class the package deals with is best described by so-called structured additive regression (STAR) models, where some characteristic ξ of the conditional distribution of a response variable Y given features X is modeled through a regression function f of the features $\xi(Y|X = x) = f(x)$. In order to facilitate parsimonious and interpretable models, the regression function f is structured, that is, restricted to additive functions $f(x) = \sum_{j=1}^{p} f_j(x)$. Each model component $f_j(x)$ might take only

a subset of the features into account. Special cases are linear models $f(x) = x^\top \beta$, additive models $f(x) = \sum_{j=1}^{p} f_j(x^{(j)})$, where f_j is a function of the jth feature $x^{(j)}$ only (smooth functions or stumps, for example) or a more complex function where $f(x)$ is implicitly defined as the sum of multiple decision trees including higher-order interactions. The latter case corresponds to boosting with trees. Combinations of these structures are also possible. The most important advantage of such a decomposition of the regression function is that each component of a fitted model can be looked at and interpreted separately for gaining a better understanding of the model at hand.

The characteristic ξ of the distribution depends on the measurement scale of the response Y and the scientific question to be answered. For binary or numeric variables, some function of the expectation may be appropriate, but also quantiles or expectiles may be interesting. The definition of ξ is determined by defining a loss function ρ whose empirical risk is to be minimized under some algorithmic constraints (i.e., limited number of boosting iterations). The model is then fitted using

$$(\hat{f}_1, \ldots, \hat{f}_p) = \operatorname*{argmin}_{(f_1,\ldots,f_p)} \sum_{i=1}^{n} w_i \rho \left(y_i, \sum_{j=1}^{p} f_j(x) \right).$$

Here $(y_i, x_i), i = 1, \ldots, n$, are n training samples with responses y_i and potentially high-dimensional feature vectors x_i, and w_i are some weights. The component-wise boosting algorithm starts with some offset for f and iteratively fits residuals defined by the negative gradient of the loss function evaluated at the current fit by updating only the best model component in each iteration. The details have been described by Bühlmann and Yu (2003). Early stopping via resampling approaches or AIC leads to sparse models in the sense that only a subset of important model components f_j defines the final model. A more thorough introduction to boosting with applications in statistics based on version 1.0 of *mboost* is given by Bühlmann and Hothorn (2007).

As of version 2.0, the package allows for fitting models to binary, numeric, ordered and censored responses, that is, regression of the mean, robust regression, classification (logistic and exponential loss), ordinal regression,[1] quantile[1] and expectile[1] regression, censored regression (including Cox, Weibull[1], log-logistic[1] or lognormal[1] models) as well as Poisson and negative binomial regression[1] for count data can be performed. Because the structure of the regression function $f(x)$ can be chosen independently from the loss function ρ, interesting new models can be fitted (e.g., in geoadditive regression, Kneib et al., 2009).

2. Design and Implementation

The package incorporates an infrastructure for representing loss functions (so-called 'families'), base-learners defining the structure of the regression function and thus the model components f_j, and a generic implementation of component-wise functional gradient descent. The main progress in version 2.0 is that only one implementation of the boosting algorithm is applied to all possible models (linear, additive, tree-based) and all families. Earlier versions were based on three implementations, one for linear models, one for additive models, and one for tree-based boosting. In comparison to the 1.0 series, the reduced code basis is easier to maintain, more robust and regression tests have been set-up in a more unified way. Specifically, the new code basis results in an enhanced and more user-friendly formula interface. In addition, convenience functions for hyper-parameter selection, faster computation of predictions and improved visual model diagnostics are available.

1. Model family is new in version 2.0 or was added after the release of *mboost* 1.0.

Currently implemented base-learners include component-wise linear models (where only one variable is updated in each iteration of the algorithm), additive models with quadratic penalties (e.g., for fitting smooth functions via penalized splines, varying coefficients or bi- and trivariate tensor product splines, Schmid and Hothorn, 2008), and trees.

As a major improvement over the 1.0 series, computations on larger data sets (both with respect to the number of observations and the number of variables) are now facilitated by memory efficient implementations of the base-learners, mostly by applying sparse matrix techniques (package *Matrix*, Bates and Mächler, 2009) and parallelization for a cross-validation-based choice of the number of boosting iterations (per default via package *multicore*, Urbanek, 2009). A more elaborate description of *mboost* 2.0 features is available from the `mboost` vignette.[2]

3. User Interface by Example

We illustrate the main components of the user-interface by a small example on human body fat composition: Garcia et al. (2005) used a linear model for predicting body fat content by means of common anthropometric measurements that were obtained for $n = 71$ healthy German women. In addition, the women's body composition was measured by Dual Energy X-Ray Absorptiometry (DXA). The aim is to describe the DXA measurements as a function of the anthropometric features. Here, we extend the linear model by i) an intrinsic variable selection via early stopping, ii) additional terms allowing for smooth deviations from linearity where necessary (by means of penalized splines orthogonalized to the linear effect, Kneib et al., 2009), iii) a possible interaction between two variables with known impact on body fat composition (hip and waist circumference) and iv) using a robust median regression approach instead of L_2 risk. For the data (available as data frame `bodyfat`), the model structure is specified via a formula involving the base-learners corresponding to the different model components (linear terms: `bols()`; smooth terms: `bbs()`; interactions: `btree()`). The loss function (here, the check function for the 0.5 quantile) along with its negative gradient function are defined by the `QuantReg(0.5)` family (Fenske et al., 2009). The model structure (specified using the formula `fm`), the data and the family are then passed to function `mboost()` for model fitting:[3]

```
R> library("mboost")                ### attach package 'mboost'
R> print(fm)                        ### model structure

DEXfat ~ bols(age) + bols(waistcirc) + bols(hipcirc) + bols(elbowbreadth) +
    bols(kneebreadth) + bols(anthro3a) + bols(anthro3b) + bols(anthro3c) +
    bols(anthro4) + bbs(age, center = TRUE, df = 1) + bbs(waistcirc,
    center = TRUE, df = 1) + bbs(hipcirc, center = TRUE, df = 1) +
    bbs(elbowbreadth, center = TRUE, df = 1) + bbs(kneebreadth,
    center = TRUE, df = 1) + bbs(anthro3a, center = TRUE, df = 1) +
    bbs(anthro3b, center = TRUE, df = 1) + bbs(anthro3c, center = TRUE,
    df = 1) + bbs(anthro4, center = TRUE, df = 1) + btree(hipcirc,
    waistcirc, tree_controls = ctree_control(maxdepth = 2, mincriterion = 0))

R> ### fit model for conditional median of DEXfat
R> model <- mboost(fm,                          ### model structure
```

2. Accessible via `vignette("mboost", package = "mboost")`.

3. The complete R code for reproducing this example is given in the accompanying file `mboost-MLOSS.R`.

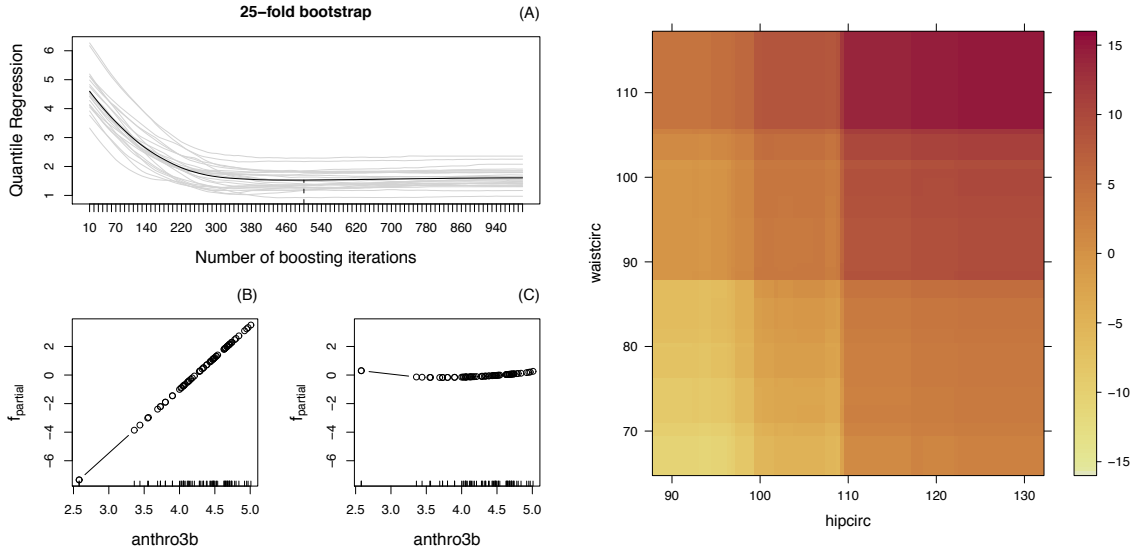


Figure 1: Out-of-bag empirical risk (A) indicating that 500 iterations are appropriate. Fitted model components for variable `anthro3b`, consisting of a linear (B) and smooth term (C). The right panel shows the interaction model component between hip and waist circumferences.

```
+                     data = bodyfat,                ### 71 observations
+                     family = QuantReg(tau = 0.5)) ### median regression
```

Once the model has been fitted it is important to assess the appropriate number of boosting iterations via the out-of-sample empirical risk. By default, 25 bootstrap samples from the training data are drawn and the out-of-bag empirical risk is computed (parallel computation if possible):

```
R> ### bootstrap for assessing the 'optimal' number
R> ### of boosting iterations
R> cvm <- cvrisk(model, grid = 1:100 * 10)
R> model[mstop(cvm)] ### restrict model to optimal mstop(cvm) iterations
```

Now, the final model is ready for a visual inspection:

```
R> plot(cvm)   ### depict out-of bag risk and
R> plot(model) ### selected components
```

The resulting plots are given in Figure 1. They indicate that a model based on three components, including a smooth function of `anthro3b` and a bivariate function of hip and waist circumference, provides the best characterization of the median body fat composition (given the model specification offered to the boosting algorithm). A hip circumference larger than 110 cm leads to increased body fat but only if the waist circumference is larger than 90 cm.

The sources of the *mboost* package are distributed at the Comprehensive R Archive Network under GPL-2, along with binaries for all major platforms as well as documentation and regression tests. Development versions are available from `http://R-forge.R-project.org`.

References

Douglas Bates and Martin Mächler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2009. URL http://CRAN.R-project.org/package=Matrix. R package version 0.999375-38.

Peter Bühlmann and Torsten Hothorn. Boosting algorithms: Regularization, prediction and model fitting (with discussion). *Statistical Science*, 22(4):477–505, 2007.

Peter Bühlmann and Bin Yu. Boosting with the L_2 loss: Regression and classification. *Journal of the American Statistical Association*, 98:324–339, 2003.

Nora Fenske, Thomas Kneib, and Torsten Hothorn. Identifying risk factors for severe childhood malnutrition by boosting additive quantile regression. Technical report, Institut für Statistik, Ludwig-Maximilians-Universität München, 2009. URL http://epub.ub.uni-muenchen.de/10510/.

Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion). *The Annals of Statistics*, 28:337–407, 2000.

Ada L. Garcia, Karen Wagner, Torsten Hothorn, Corinna Koebnick, Hans-Joachim F. Zunft, and Ulrike Trippo. Improved prediction of body fat by measuring skinfold thickness, circumferences, and bone breadths. *Obesity Research*, 13(3):626–634, 2005.

Torsten Hothorn, Peter Bühlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. *mboost: Model-Based Boosting*, 2010. URL http://CRAN.R-project.org/package=mboost. R package version 2.0-4.

Thomas Kneib, Torsten Hothorn, and Gerhard Tutz. Variable selection and model choice in geoadditive regression models. *Biometrics*, 65(2):626–634, 2009.

Matthias Schmid and Torsten Hothorn. Boosting additive models using component-wise P-splines as base-learners. *Computational Statistics & Data Analysis*, 53(2):298–311, 2008.

Simon Urbanek. *multicore: Parallel Processing of R Code on Machines with Multiple Cores or CPUs*, 2009. URL http://www.rforge.net/multicore/. R package version 0.1-3.

Journal of Machine Learning Research 11 (2010) 2115-2140 Submitted 3/09; Revised 6/10; Published 8/10

Importance Sampling for Continuous Time Bayesian Networks

Yu Fan YFAN@CS.UCR.EDU
Jing Xu JINGXU@CS.UCR.EDU
Christian R. Shelton CSHELTON@CS.UCR.EDU
Department of Computer Science and Engineering
University of California
Riverside, CA, 92521, USA

Editor: Carl Edward Rasmussen

Abstract

A continuous time Bayesian network (CTBN) uses a structured representation to describe a dynamic system with a finite number of states which evolves in continuous time. Exact inference in a CTBN is often intractable as the state space of the dynamic system grows exponentially with the number of variables. In this paper, we first present an approximate inference algorithm based on importance sampling. We then extend it to continuous-time particle filtering and smoothing algorithms. These three algorithms can estimate the expectation of any function of a trajectory, conditioned on any evidence set constraining the values of subsets of the variables over subsets of the time line. We present experimental results on both synthetic networks and a network learned from a real data set on people's life history events. We show the accuracy as well as the time efficiency of our algorithms, and compare them to other approximate algorithms: expectation propagation and Gibbs sampling.

Keywords: continuous time Bayesian networks, importance sampling, approximate inference, filtering, smoothing

1. Introduction

Many real world applications involve highly complex dynamic systems. These systems usually contain a large number of stochastic variables, which evolve asynchronously in continuous time. Such dynamic systems include computer networks, sensor networks, social networks, mobile robots, and cellular metabolisms. Modeling, learning and reasoning about these complex dynamic systems is an important task and a great challenge.

1.1 Structured Process Representation

A central task of the above applications is to calculate probability distributions of the system over time. For instance, we may wish to know the distribution over when a variable will change next or the state of a current variable, given past (partial) evidence. However, as the number of the variables increases, the state space of the distribution grows exponentially. Such growth makes the inference task very difficult for large systems. One solution is to use structured representation to factorize the state space according to the dependencies of the variables. For dynamic systems, Dynamic Bayesian Networks (DBNs) (Dean and Kanazawa, 1989) are commonly used. A DBN describes the dynamic system as a time-sliced model by measuring the evolution of the system with a (usually fixed) time interval Δt . The transition probabilities from states at time t to states at time $t + \Delta t$ are represented by a Bayesian network. DBNs can work well for systems that are observed at regular time steps. However, for many applications, discretizing time has several limitations. First, we usually choose a fixed time interval, Δt. In many real world systems, variables evolve at different time granularities. Some variables may evolve very fast whereas some evolve very slowly. Choosing an appropriate time interval is a difficult task. Larger Δt may result in an inaccurate model while smaller Δt may cause inference in the model to be inefficient.

Second, the dependencies of the transition model are unstable with respect to Δt. That is, different choices of Δt may result in different network structures between t and $t + \Delta t$. The network structure represents independencies between variables at t and Δt. This is a function of Δt, both theoretically and empirically (Nodelman et al., 2003). If Δt is an inherent parameter of the process, this is not a problem. However, if it is chosen for estimation or computational reasons, this becomes an issue as its choice is not unique. Finally, DBNs (and discrete-time Markov processes in general) do not necessarily correspond to processes that are Markovian outside of the sampled instants of time. Consider that if T is the transition matrix for a process with time interval Δt, $T^{1/2}$ is the transition matrix for the same process with time interval $\frac{\Delta t}{2}$. However, such a square root may not exist in the space of real matrices. Therefore, there may not be any simple extension of a DBN to the times between the sampled instants.

An alternative and more natural approach to model dynamic systems is to use a continuous-time model. For systems with a finite number of states, one way is to consider the entire system as a continuous-time discrete-state Markov process. Like discrete-time processes, this method suffers from the fact that the state space of the process grows exponentially with the number of variables in the system. Recently, Nodelman et al. (2002) extended this framework to a *continuous time Bayesian network* (CTBN), which factorizes a system into local variables using a graphical representation, much as a DBN does for a discrete-time process. Parameter estimation in CTBNs with fully observed data and partially observed data were provided in Nodelman et al. (2003) and Nodelman et al. (2005b) respectively. Because CTBNs explicitly represent the temporal dynamics in continuous time and explore the dependencies among stochastic variables using a structured representation, they have been applied to various real world systems, including human-computer interactions (Nodelman and Horvitz, 2003), server farm failures (Herbrich et al., 2007), robot monitoring (Ng et al., 2005) and network intrusion detection (Xu and Shelton, 2008). Kan and Shelton (2008) used the CTBN representation in their solution of structured continuous-time Markov decision processes.

Queueing theory (Bolch et al., 1998) and Petri nets (Petri, 1962) provide an alternative continuous-time structured process models. However, they make different assumptions about the structure. They were designed to answer questions about steady-state distributions. Their algorithms are not suited to learning from partial data nor to answering many statistical questions. A singular and recent exception is the work of Sutton and Jordan (2008) which applied Gibbs sampling to queueing models.

1.2 Prior CTBN Inference Methods

In CTBNs, a trajectory (or sample) consists of the starting values for the system along with the (real-valued) times at which the variables change and their corresponding new values. Inference for a CTBN is the task of estimating the distribution over trajectories given a partial trajectory (in which some values or transitions are missing for some variables during some time intervals). Inference plays a central role as it not only helps us answer queries about distributions, but it is also involved in parameter estimation when the observation data is incomplete. Performing exact inference in a CTBN requires constructing a single rate matrix for the entire system and computing the exponential of the matrix, which is often intractable: the exponentiation must be performed separately for each period of constant evidence and (more problematic) even a sparse representation of the matrix may not fit in memory. Thus, many applications of CTBNs require an approximate inference method. A method based on expectation propagation (Minka, 2001) was presented in Nodelman et al. (2005a). Saria et al. (2007) extended it to full belief propagation and provided a method to adapt the approximation quality.

Other approximate inference methods are based on sampling. They have the advantage of being anytime algorithms. (We can stop at any time during the computation and obtain an answer.) Furthermore, in the limit of infinite samples (computation time), they converge to the true answer.

As we note below, because time is a continuous variable, any evidence containing a record of the change in a variable has a zero probability under the model. Therefore rejection sampling and straightforward likelihood weighting are generally not viable methods.

Ng et al. (2005) developed a continuous-time particle filtering algorithm. However, it only handles point evidence on binary and ternary discrete variables using rejection sampling and focuses primarily on the incor-

poration of evidence from a continuous-state part of the system (which we do not consider here). Recently, El-Hay et al. (2008) provided another sampling algorithm for CTBNs using Gibbs sampling. The algorithm starts from an arbitrary trajectory that is consistent with the evidence. Then, in each iteration, it randomly picks one variable X, and samples an entire trajectory for that variable by fixing the trajectory of all the other variables. Since only X is not fixed, the conditioned cumulative distribution that X stays in one state less than t and the state transition probabilities can be calculated exactly using standard forward and backward propagation within the Markov blanket of X. The Gibbs sampling algorithm can handle any type of evidence and it provides an approach to sample from the exact posterior distribution given the evidence. However, the posterior distribution can be any arbitrary function. To sample exactly from it, binary search has to be applied and $F(t)$ is repeatedly evaluated, which may affect the efficiency of the algorithm.

1.3 Outline of This Work

In this paper we explore a different sampling approach using importance sampling. Our algorithm generates weighted samples to approximate the expectation of a function of the trajectory. It differs from previous approaches in a number of key ways. There is no exact inference method involved in our approach. Thus, our algorithm does not depend on complex numeric computations. The transition times for variables are sampled from regular exponential distributions in our algorithm, which can be done in constant time. Our algorithm can be adapted to a population-based filter (a particle filter). It can handle both point and continuous evidence, is simple to implement, and can be easily extended to continuous time systems other than CTBNs. The formulation of this sampling procedure is not trivial due to the infinite extent of the trajectory space, both in the transition time continuum and the number of transitions. The algorithm was first presented in Fan and Shelton (2008). This paper extends that work by comparing the algorithm to the newly developed Gibbs sampling algorithm (El-Hay et al., 2008), evaluating its performance on parameter learning with partially observed data, and demonstrating its performance on real-world networks.

The remainder of the paper is structured as follows. In Section 2, we briefly describe the notation for CTBNs. In Section 3, we describe our importance sampling algorithm for CTBNs and extend the algorithm to particle filtering and particle smoothing algorithms. In Section 4, we describe our experiment results.

2. Continuous Time Bayesian Networks

We first briefly describe the definition, likelihood, and sufficient statistics of the CTBN model. We then review the exact inference and parameter estimation algorithms for CTBNs.

2.1 The CTBN Model

Continuous time Bayesian networks (Nodelman et al., 2002) are based on the framework of continuous time, finite state, homogeneous Markov processes (Norris, 1997). Let X be a continuous time, finite state, homogeneous Markov process with n states $\{x_1,\ldots,x_n\}$. The behavior of X is described by the initial distribution P_X^0 and the transition model which is often represented by the intensity matrix

$$\mathbf{Q}_X = \begin{bmatrix} -q_{x_1} & q_{x_1x_2} & \cdots & q_{x_1x_n} \\ q_{x_2x_1} & -q_{x_2} & \cdots & q_{x_2x_n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{x_nx_1} & q_{x_nx_2} & \cdots & -q_{x_n} \end{bmatrix},$$

where $q_{x_ix_j}$ is the intensity with which X transitions from x_i to x_j and $q_{x_i} = \sum_{j\neq i} q_{x_ix_j}$. The intensity matrix $\mathbf{Q}_X$ is time invariant. Given $\mathbf{Q}_X$, the transient behavior of X can be described as the following: X stays in state x_i for an amount of time t and transitions to state x_j. t is exponentially distributed with parameter q_{x_i}. That is, the probability density function and the corresponding distribution function for X staying in state x_i

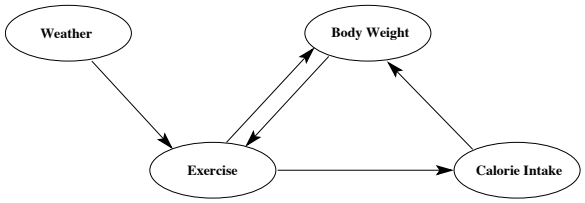


Figure 1: CTBN Example: Weight Control Effect

are

$$f(q_{x_i},t) = q_{x_i}\exp(-q_{x_i}t), \quad t \geq 0.$$
$$F(q_{x_i},t) = 1-\exp(-q_{x_i}t), \quad t \geq 0.$$

The expected time of transitioning is $1/q_{x_i}$. Upon transitioning, the probability X transitions from state x_i to x_j is $\theta_{x_i x_j} = q_{x_i x_j}/q_{x_i}$.The distribution over the state of X at time t can be calculated as

$$P_X(t) = P_X^0 \exp(\mathbf{Q}_X t)$$

where P_X^0 is the distribution over X at time 0 represented as a row vector, and exp is the matrix exponential.

To model a dynamic system containing several variables, we can consider the whole system as one variable, enumerate the entire state space, calculate the transition intensity of each pair of these states and put them into a single intensity matrix. However, the size of the state space grows exponentially with the number of variables in the system, which makes this method infeasible for large systems.

Nodelman et al. (2002) defined a *continuous time Bayesian network* (CTBN), which uses a graphical model to provide a compact factored representation of continuous time Markov process. A CTBN models each local variable X as an inhomogeneous Markov process, whose parametrization depends on some subset of other variables $\mathbf{U}$. The intensity matrix of X is called a conditional intensity matrix (*CIM*) $\mathbf{Q}_{X|\mathbf{U}}$, which is defined as a set of intensity matrices $\mathbf{Q}_{X|\mathbf{u}}$, one for each instantiation $\mathbf{u}$ of the variable set $\mathbf{U}$. The evolution of X depends instantaneously on the values of the variables in $\mathbf{U}$.

Let $\boldsymbol{X}$ be a dynamic system containing several variables X. A *continuous time Bayesian network* $\mathcal{N}$ over $\boldsymbol{X}$ consists of two components: an *initial distribution* $P_{\boldsymbol{X}}^0$, specified as a Bayesian network $\mathcal{B}$ over $\boldsymbol{X}$, and a *continuous transition model*, specified using a directed (possibly cyclic) graph $\mathcal{G}$ whose nodes are $X \in \boldsymbol{X}$. Let $\mathbf{U}_X$ denote the parents of X in $\mathcal{G}$. Each variable $X \in \boldsymbol{X}$ is associated with a conditional intensity matrix, $\mathbf{Q}_{X|\mathbf{U}_X}$.

Example 1 *Assume we want to model the behavior of a person controlling his body weight. When the person is overweight, he may exercise more to lose the excess weight. Increasing exercise intensity tends to increase his appetite, which will increase his daily calorie intake. Both exercise intensity and calorie intake contribute to his body weight. Furthermore, the exercise intensity also depends on the weather. Such a dynamic system contains four variables: body weight, exercise, calorie intake, and weather. Each variable changes in continuous time and its change rate depends on the current value of some other variables.*

We can use a CTBN to represent such behavior. The dependencies of these four variables are depicted using a graphical structure, as shown in Figure 1. The quantitative transient dynamics for each variable is represented using a conditional intensity matrix. Let us assume all the four variables are binary. Let $B(t)$ be the person's body weight ($Val(B(t)) = \{b_0 = normal, b_1 = overweight\}$), $E(t)$ be the exercise intensity ($Val(E(t) = \{e_0 = light, e_1 = heavy\}$), $C(t)$ be his daily calorie intake ($Val(C(t) = \{c_0 = low, c_1 = high\}$) and $W(t)$ be the weather ($Val(W(t) = \{w_0 = rainy, w_1 = sunny\}$). The conditional intensity matrices for the four variables can be specified as

$\mathbf{Q}_W$	$\mathbf{Q}_W$	$=$	$\begin{bmatrix} -0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}$,				
$\mathbf{Q}_{E\|W,B}$	$\mathbf{Q}_{E\|w_0,b_0}$	$=$	$\begin{bmatrix} -0.1 & 0.1 \\ 2 & -2 \end{bmatrix}$,	$\mathbf{Q}_{E\|w_1,b_0}$	$=$	$\begin{bmatrix} -0.3 & 0.3 \\ 1 & -1 \end{bmatrix}$,
	$\mathbf{Q}_{E\|w_0,b_1}$	$=$	$\begin{bmatrix} -0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}$,	$\mathbf{Q}_{E\|w_1,b_0}$	$=$	$\begin{bmatrix} -1 & 1 \\ 0.1 & -0.1 \end{bmatrix}$,
$\mathbf{Q}_{C\|E}$	$\mathbf{Q}_{C\|e_0}$	$=$	$\begin{bmatrix} -0.2 & 0.2 \\ 1 & -1 \end{bmatrix}$,	$\mathbf{Q}_{C\|e_1}$	$=$	$\begin{bmatrix} -1 & 1 \\ 0.2 & -0.2 \end{bmatrix}$,
$\mathbf{Q}_{B\|E,C}$	$\mathbf{Q}_{B\|e_0,c_0}$	$=$	$\begin{bmatrix} -0.2 & 0.2 \\ 0.8 & -0.8 \end{bmatrix}$,	$\mathbf{Q}_{B\|e_1,c_0}$	$=$	$\begin{bmatrix} -0.1 & 0.1 \\ 1 & -1 \end{bmatrix}$,
	$\mathbf{Q}_{B\|e_0,c_1}$	$=$	$\begin{bmatrix} -1 & 1 \\ 0.1 & -0.1 \end{bmatrix}$,	$\mathbf{Q}_{B\|e_1,c_1}$	$=$	$\begin{bmatrix} -0.2 & 0.2 \\ 0.6 & -0.6 \end{bmatrix}$.

Notice that unlike Bayesian networks, the CTBN model allows cycles. The transient behavior of each local variable is controlled by the current value of its parents. If the person is doing light exercise and his calorie intake is low, the dynamics of his body weight are determined by the intensity matrix $\mathbf{Q}_{B|e_0,c_0}$*. If the time unit is one month, we expect his weight will go back to normal in* $1/0.8 = 1.25$ *months if he is currently overweight and doing light exercise and controlling his daily calorie intake.*

We can also use a single continuous time Markov process to represent this network, which requires an intensity matrix of size 16×16*. To generate the single intensity matrix, we can follow the amalgamation algorithm in Nodelman et al. (2002). Basically, we enumerate the entire state space* (W,E,C,B)*, and assign intensity 0 to transitions that change two variables simultaneously. For any transition involving only one of the variables, simply use the entry from the appropriate intensity matrix above. The resulting matrix is*

$$
\begin{array}{c}
\\
w_0e_0c_0b_0 \\ w_1e_0c_0b_0 \\ w_0e_1c_0b_0 \\ w_1e_1c_0b_0 \\ w_0e_0c_1b_0 \\ w_1e_0c_1b_0 \\ w_0e_1c_1b_0 \\ w_1e_1c_1b_0 \\ w_0e_0c_0b_1 \\ w_1e_0c_0b_1 \\ w_0e_1c_0b_1 \\ w_1e_1c_0b_1 \\ w_0e_0c_1b_1 \\ w_1e_0c_1b_1 \\ w_0e_1c_1b_1 \\ w_1e_1c_1b_1
\end{array}
\begin{bmatrix}
\begin{array}{cccccccccccccccc}
w_0e_0c_0b_0 & w_1e_0c_0b_0 & w_0e_1c_0b_0 & w_1e_1c_0b_0 & w_0e_0c_1b_0 & w_1e_0c_1b_0 & w_0e_1c_1b_0 & w_1e_1c_1b_0 & w_0e_0c_0b_1 & w_1e_0c_0b_1 & w_0e_1c_0b_1 & w_1e_1c_0b_1 & w_0e_0c_1b_1 & w_1e_0c_1b_1 & w_0e_1c_1b_1 & w_1e_1c_1b_1 \\
-1 & 0.5 & 0.1 & 0 & 0.2 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.5 & -1.2 & 0 & 0.3 & 0 & 0.2 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & -3.6 & 0.5 & 0 & 0 & 1 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0.5 & -2.6 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & -2.6 & 0.5 & 0.1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0.5 & -2.8 & 0 & 0.3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0.2 & 0 & 2 & 0 & -2.9 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 \\
0 & 0 & 0 & 0.2 & 0 & 1 & 0.5 & -1.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 \\
0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0.5 & 0.5 & 0 & 0.2 & 0 & 0 & 0 \\
0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & -2.5 & 0 & 1 & 0 & 0.2 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & -3 & 0.5 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.5 & -2.6 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2.1 & 0.5 & 0.5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 1 & 0 & 0 & 0.5 & -2.6 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0.2 & 0 & 0.5 & 0 & -1.8 & 0.5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0.2 & 0 & 0.1 & 0.5 & -1.4
\end{array}
\end{bmatrix}.
$$

As we include more variables in this system, the size of the intensity matrix grows exponentially with the number of variables.

2.2 Likelihood and Sufficient Statistics

The probability density over trajectories σ of a set of variables $\boldsymbol{X}$ described by a CTBN belongs to the exponential family. Therefore, the distribution of a CTBN can be described in terms of the sufficient statistics

of σ (Nodelman et al., 2003). Let $T[x|\mathbf{u}]$ be the amount of time $X = x$ while $\mathbf{U}_X = \mathbf{u}$, and $M[x,x'|\mathbf{u}]$ be the number of transitions from x to x' while $\mathbf{U}_X = \mathbf{u}$. If we let $M[x|\mathbf{u}] = \sum_{x'} M[x,x'|\mathbf{u}]$, the probability density of trajectory σ (omitting the starting distribution) is

$$P_{\mathcal{N}}(\sigma) = \prod_{X \in \mathbf{X}} L_X(T[X|\mathbf{U}_X], M[X|\mathbf{U}_X]) \tag{1}$$

where

$$L_X(T[X|\mathbf{U}_X], M[X|\mathbf{U}_X]) = \prod_{\mathbf{u}} \prod_{x} \left(q_{x|\mathbf{u}}^{M[x|\mathbf{u}]} \exp(-q_{x|\mathbf{u}} T[x|\mathbf{u}]) \prod_{x' \neq x} \theta_{xx'|\mathbf{u}}^{M[x,x'|\mathbf{u}]} \right) \tag{2}$$

is the local likelihood for variable X. The likelihood also decomposes by time. That is, the likelihood of a trajectory on $[0,T)$ is equal to the likelihood based only on sufficient statistics from time 0 to time t multiplied by the likelihood based only on sufficient statistics from time t to time T.

2.3 Evidence and Queries

Given a CTBN model, we would like to use it to answer queries conditioned on observations. There are two common types of observations: point evidence and continuous evidence. Point evidence represents the observation of the value of some variables at a particular time instant. Continuous evidence provides the behavior of some variables throughout an interval $[t_1,t_2)$. For instance, $x = 1$ during the interval $[2,3.5)$, or $x = 1$ from $t = 2$ to $t = 3$ and then x transitions to $x = 0$ at $t = 3$ and stays in that state until $t = 5$. We define $x[t_1 : t_2)$ be the behavior of variable X on the interval $[t_1,t_2)$, $x[t_1 : t_2]$ be the behavior of X on the interval $[t_1,t_2]$ and $x(t_1 : t_2]$ be the behavior of X on the interval $(t_1,t_2]$.

Queries can ask about the marginal distribution of some variables at a particular time, such as the distribution of x and y at $t = 2$, or questions about the timing of a transition, such as the distribution over the time that y transitions from $y = 1$ to $y = 2$ for the first time in the interval $[1,4)$. In learning (especially when employing expectation-maximization), we might query the expected sufficient statistics of a CTBN, which include the total amount of time that a variable spends in a state, and the total number of times that a variable transitions from one state to another state under certain conditions. For example, we might want to know the total amount of time that $x = 0$ throughout the entire interval, or the number of times that x transitions from 1 to 2 during the time interval $[2,3)$ when $y = 0$. In this paper, we will concentrate on answering queries given the continuous evidence, but our method can be trivially extended to point evidence.

2.4 Exact Inference in CTBNs

A CTBN can be viewed as a homogeneous Markov process with a large joint intensity matrix amalgamated from the CIMs of the CTBN. Exact inference in a CTBN can be performed by generating a single joint intensity matrix over the entire state space of the CTBN and running the forward-backward algorithm on the joint intensity matrix of the homogeneous Markov process. We review this method here, but a more complete treatment can be found in Nodelman et al. (2002).

Assume that we have a partially observed trajectory σ of a CTBN $\mathcal{N}$ from 0 to T. We can divide the evidence σ into N intervals $[t_i,t_{i+1})$ $(i = 0,\ldots,N-1)$ according to the observed transition times. That is, each interval contains a constant observation of the CTBN, and t_i is the time that a variable begins to be observed, stops being observed, or is observed to transition. We set $t_0 = 0$ and $t_N = T$.

To perform exact inference, we first generate the intensity matrix $\mathbf{Q}$ for the joint homogeneous Markov process and incorporate the evidence into $\mathbf{Q}$. If each variable X_i in the CTBN $\mathcal{N}$ has n_i states, the number of states of the joint Markov process is $n = \prod n_i$ and $\mathbf{Q}$ is an $n \times n$ matrix. The value of the off-diagonal element q_{ij} in $\mathbf{Q}$ for which only one variable value is different between states i and j is the corresponding intensity in the CIM of that variable. All the other off-diagonal elements are zero since two variables can not transition at exactly the same time in a CTBN. The diagonal elements are computed to make each row sum to zero.

To incorporate the evidence, we reduce the joint intensity matrix $\mathbf{Q}$ to $\mathbf{Q}_i$ for each interval $[t_i,t_{i+1})$ by zeroing out the rows and columns of $\mathbf{Q}$ which represent states that are inconsistent with the evidence. Addi-

tionally, let $\mathbf{Q}_{i,j}$ be the matrix $\mathbf{Q}$ with all elements zeroed out except the off-diagonal elements that represent the intensities of transitioning from non-zero rows in $\mathbf{Q}_i$ to non-zero columns in $\mathbf{Q}_j$. If evidence blocks i and j differs only in which variables are observed (no transition is observed between them), then $\mathbf{Q}_{i,j}$ is the identity matrix instead.

$\exp(\mathbf{Q}_i(t_{i+1}-t_i))$ represents the transition matrix for interval $[t_i,t_{i+1})$ and $\mathbf{Q}_{i,i+1}$ corresponds to the transition probability density between two consecutive intervals at time t_{i+1}. We can use the forward-backward algorithm for Markov process to answer queries.

We define the forward and backward probability vectors $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ as

$$\begin{aligned}\boldsymbol{\alpha}_t &= p(X_t, \sigma_{[0,t)}),\\ \boldsymbol{\beta}_t &= p(\sigma_{[t,T)}|X_t)\ .\end{aligned}$$

Let $\boldsymbol{\alpha}_0$ be the initial distribution P_X^0 over the state and $\boldsymbol{\beta}_T$ be a vector of ones. The forward and backward distribution vector for each interval can be calculated recursively:

$$\begin{aligned}\boldsymbol{\alpha}_{t_{i+1}} &= \boldsymbol{\alpha}_{t_i}\exp(\mathbf{Q}_i(t_{i+1}-t_i))\mathbf{Q}_{i,i+1},\\ \boldsymbol{\beta}_{t_i} &= \mathbf{Q}_{i-1,i}\exp(\mathbf{Q}_i(t_{i+1}-t_i))\boldsymbol{\beta}_{t_{i+1}}\ .\end{aligned}$$

The distribution over the state of the CTBN at time $t \in [t_i,t_{i+1})$ given the evidence $\sigma_{[0,T)}$ can be computed as

$$P(X_t = k, \sigma_{[0,T)}) = \boldsymbol{\alpha}_{t_i}\exp(\mathbf{Q}_i(t-t_i))\boldsymbol{\Delta}_{k,k}\exp(\mathbf{Q}_i(t_{i+1}-t))\boldsymbol{\beta}_{t_{i+1}} \tag{3}$$

where $\boldsymbol{\Delta}_{i,j}$ is an $n \times n$ matrix of zeros with a single one in position i,j. Other queries can be similarly computed.

2.5 CTBN Parameter Estimation

Given a set of trajectories $D = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ and a fixed graphical structure, we would like to estimate the parameters (the conditional intensity matrix) of the CTBN model.

When the data set D is complete, where each trajectory σ_i is a complete set of state transitions and the times at which they occurred, the parameters can be learned by maximizing the log-likelihood of the data set (Nodelman et al., 2003). According to Equation 1 and Equation 2, the log-likelihood can be written as the sum of the log-likelihood for each local variable. By maximizing the log-likelihoods, the parameters can be derived as

$$\hat{q}_{x|\mathbf{u}} = \frac{M[x|\mathbf{u}]}{T[x|\mathbf{u}]}\ ; \quad \hat{\theta}_{xx'|\mathbf{u}} = \frac{M[x,x'|\mathbf{u}]}{M[x|\mathbf{u}]}\ . \tag{4}$$

When the data set is incomplete, the expectation maximization (EM) algorithm (Dempster et al., 1977) can be used to find the maximum likelihood parameters (Nodelman et al., 2005b). The EM algorithm begins with an arbitrary initial parameter assignment, and alternatively repeats the expectation step and maximization step until convergence. In expectation step, for each trajectory $\sigma_i \in D$, expected sufficient statistics $\bar{M}[x|\mathbf{u}]$, $\bar{M}[x,x'|\mathbf{u}]$ and $\bar{T}[x|\mathbf{u}]$ are computed using exact inference. In maximization step, new parameters are computed according to Equation 4 as if the expected sufficient statistics came from complete data.

3. Sampling-based Inference

As we described in the previous section, exact inference in a CTBN can be performed by generating a single joint intensity matrix over the entire state space. As the number of states is exponential in the number of the nodes in the network, this approach is infeasible when the network size is large. In this section we describe an algorithm for approximate CTBN inference based on importance sampling.

Procedure *CTBN-Sample*(t_{end})
1. $t \leftarrow 0, \sigma \leftarrow \emptyset$
2. For each variable $X \in \boldsymbol{X}$
   Choose state $x(0)$ according to $\boldsymbol{\theta}^{\mathcal{B}}_{X|\mathbf{pa}_{\mathcal{B}}(X)}$.

Loop:

3. For each variable X such that *Time*(X) is undefined:
   Choose Δt for next X transition from an exponential with parameter $q_{x(t)|\mathbf{u}_X(t)}$.
   Define *Time*(X) $\leftarrow t + \Delta t$
4. Let $X = \arg\min_{X \in \boldsymbol{X}} [\textit{Time}(X)]$
5. If *Time*(X) $\geq t_{end}$ **return** σ
6. Update $t \leftarrow$ *Time*(X)
7. Choose $x(t)$, the next value of X, from the multinomial with parameters $\boldsymbol{\theta}_{x(t)|\mathbf{u}_X(t)}$.
   Add $\langle X \leftarrow x(t), t \rangle$ to σ.
   Undefine *Time*(X), and *Time*(Y) for all variables Y for which $X \in \mathbf{U}_Y$.

Figure 2: Forward sampling semantics for a CTBN

3.1 Forward Sampling

Queries that are not conditioned on evidence can be answered by randomly sampling many trajectories and looking at the fraction that match the query. More formally, if we have a CTBN $\mathcal{N}$ we generate a set of particles $\mathcal{D} = \{\sigma[1], \ldots, \sigma[M]\}$ where each particle is a sampled trajectory. With $\mathcal{D}$ we can estimate the expectation of any function g by computing

$$\hat{\mathbf{E}}_{\mathcal{N}}[g] = \frac{1}{M} \sum_{m=1}^{M} g(\sigma[m]) \; . \tag{5}$$

For example, if we let $g = \mathbf{1}\{x(5) = x_1\}$ then we could use the above formula to estimate $P_{\mathcal{N}}(x(5) = x_1)$. Or the function $g(\sigma)$ might count the total number of times that X transitions from x_1 to x_2 while its parent U has value u_1, allowing us to estimate the expected sufficient statistic $M[x_1, x_2 | u_1]$. The algorithm for sampling a trajectory is shown in Figure 2. For each variable $X \in \boldsymbol{X}$, it maintains $x(t)$—the state of X at time t—and *Time*(X)—the next potential transition time for X. The algorithm adds transitions one at a time, advancing t to the next earliest variable transition. When a variable X (or one of its parents) undergoes a transition, *Time*(X) is resampled from the new exponential waiting time distribution. We use $\mathbf{u}_X(t)$ to represent the instantiation to parents of X at time t.

If we want to obtain a conditional probability of a query given evidence, the situation is more complicated. We might try to use *rejection sampling*: forward sample to generate possible trajectories, and then simply reject the ones that are inconsistent with our evidence. The remaining trajectories are sampled from the posterior distribution given the evidence, and can be used to estimate probabilities as in Equation 5. However, this approach is entirely impractical in our setting, as in any setting involving an observation of a continuous quantity—in our case, time. In particular, suppose we observe that X transitions from x_1 to x_2 at time t. The probability of sampling a trajectory in which that transition occurs at precisely that time is zero. Thus, if we have evidence about transitions, with probability 1, none of our sampled trajectories will be relevant.

3.2 Gibbs Sampling

Recently, El-Hay et al. (2008) provided a Markov Chain Monte Carlo (MCMC) procedure which used a Gibbs sampler to generate samples from the posterior distribution given the evidence.

Suppose we want to sample trajectories from a CTBN with n variables $(X_i, X_2, \ldots, X_n)$ given the evidence $\mathbf{e}$. The Gibbs sampler starts with an arbitrary trajectory that is consistent with the evidence. In each iteration, the sampler randomly picks one variable X_i and samples the entire trajectory of X_i by fixing the trajectories

of the other variables $Y = \{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n\}$ as evidence. To generate the entire trajectory of X_i according to the evidence $\mathbf{e}$, the states and transitions of X_i need to be sampled in those intervals that X_i is not observed according to the evidence. The trajectory in each unobserved interval of X_i can be generated by alternatively sampling transition time Δt and new state x from the posterior distribution given $\mathbf{e}$ and the trajectories of the other variables Y.

Assume we are sampling the trajectory of X for the interval $[0,T]$, and $X_i(0) = x_0$, $X_i(T) = x_T$. The transition time Δt is sampled by inverse transform sampling: first draw ξ from the $[0,1]$ uniform distribution and set $\Delta t = F^{-1}(\xi)$, where $F^{-1}(\xi)$ is the inverse of the conditional cumulative distribution function $F(t)$ that X_i stays in state x_0 for a time less than t:

$$F(t) = 1 - Pr(X_i(0:t] = x_0 | x_0, x_T, Y[0:T]) .$$

$F(t)$ can be calculated by decomposing $Pr(X_i(0:t] = x_0|x_0, x_T, Y[0:T])$ using the Markov property of the process:

$$Pr(X_i(0:t] = x_0|x_0, x_T, Y[0:T]) = \frac{\tilde{\alpha}(t)\tilde{\beta}_{x_0}(t)}{\tilde{\beta}_{x_0}(0)}$$

where

$$\tilde{\alpha}(t) = Pr(X_i(0:t] = x_0, Y[0:t]|x_0, Y_0),$$
$$\tilde{\beta}_x(t) = Pr(x_T, Y(t:T]|X_i(t) = x, Y(t)) .$$

$\tilde{\alpha}(t)$ and $\tilde{\beta}_x(t)$ can be calculated using a slightly modified version of the standard forward-backward algorithm described in Section 2.4. Using the fact that X_i is independent of all the other components given the entire trajectory of its Markov blanket, the computation of $\tilde{\alpha}(t)$ and $\tilde{\beta}(t)$ can be limited to X_i and its Markov blanket (the parents of X_i, the children of X_i, and the children's parents).

Since the conditional cumulative distribution function $F(t)$ can be arbitrarily complex, the inverse function $F^{-1}(t)$ can not be solved analytically. Finding Δt that satisfies $F(\Delta t) = \xi$ is performed using a two-step searching method: first find the interval $[\tau_k, \tau_{k+1}]$ that satisfies $F(\tau_k) < \xi < F(\tau_{k+1})$, where τ_k are the transition points of the Markov blanket of X_i. Then Δt is found by performing an L step binary search on the interval $[\tau_k, \tau_{k+1}]$.

The transition probability that X_i transitions from $x^{(0)}$ to a new state x can be calculated similarly:

$$Pr(X_i(t^+) = x|X_i(0:t] = x^{(0)}, Y(0:T]) = \frac{q^{X_i|Y}_{x_0,x}\tilde{\beta}_x(t)}{\sum_{x' \neq x_0} q^{X_i|Y}_{x_0,x'}\tilde{\beta}_{x'}(t)} .$$

The Gibbs sampling algorithm can handle any type of evidence. The sampled trajectories are guaranteed to be consistent with the evidence. However, sampling the transition time Δt requires using a binary search algorithm and repeatedly computing the conditional cumulative distribution function $F(t)$, which may require long running time.

3.3 Importance Sampling

In this section, we introduce another approximate inference method using importance sampling, which does not require computing the exact posterior distribution. This method first appeared in Fan and Shelton (2008).

In importance sampling, we generate samples from a proposal distribution P' which guarantees that our sampled trajectories will conform to our evidence $\mathbf{e}$. We must weight our samples to correct for the fact that we are drawing them from P' instead of the target distribution $P_{\mathcal{N}}$ defined by the CTBN. In particular, if σ is a sample from P' we set its weight to be

$$w(\sigma) = \frac{P_{\mathcal{N}}(\sigma, \mathbf{e})}{P'(\sigma)} . \tag{6}$$

In normalized importance sampling, we draw a set of samples $\mathcal{D} = \{\sigma[1], \ldots, \sigma[M]\}$ i.i.d. from the proposal distribution, and estimate the conditional expectation of a function g given evidence $\mathbf{e}$ as

$$\hat{\mathbf{E}}_{\mathcal{N}}[g \mid \mathbf{e}] = \frac{1}{W} \sum_{m=1}^{M} g(\sigma[m]) w(\sigma[m])$$

where W is the sum of the weights.

This estimator is consistent if the support of P' is a superset of the support of $P_{\mathcal{N}}$. In general, $\hat{\mathbf{E}}_{\mathcal{N}}$ is biased and the bias decreases as $O(M^{-1})$. The variance of the estimator also decreases as $O(M^{-1})$. For more information on this and related sampling estimates, see Hesterberg (1995).

For our algorithm, we base the proposal distribution on the forward sampling algorithm. As we are sampling a trajectory, we occasionally depart from the regular forward sampling algorithm and "force" the behavior of one or more variables to ensure consistency with the evidence.

3.4 Simple Evidence

The simplest query involves evidence over some subset of variables $\boldsymbol{V} \subset \boldsymbol{X}$ for the total length of the trajectory. We force only the behavior of the variables $\boldsymbol{V}$ and there are no choices about how to do that. In particular, we use the following proposal distribution: forward sample the behavior of variables $X \in (\boldsymbol{X} \setminus \boldsymbol{V})$ inserting the known transitions at known times for variables in $\boldsymbol{V}$ as determined by the evidence. As there are no choices in our forcing, the likelihood of drawing σ from the proposal distribution is just the likelihood contribution of forward sampling the behavior of the variables $X \in (\boldsymbol{X} \setminus \boldsymbol{V})$, in the context of the total behavior of the system.

According to Section 2.2, $x[t_1 : t_2)$ can be summarized by the sufficient statistics over X on the interval $[t_1, t_2)$. Let $\tilde{L}_X(x[t_1 : t_2))$ be a partial likelihood contribution function, computed by plugging the sufficient statistics of $x[t_1 : t_2)$ into Equation 2. The partial contribution function can be defined over a collection of intervals I as $\tilde{L}_X(I) = \prod_{x[t_1 : t_2) \in I} \tilde{L}_X(x[t_1 : t_2))$. Returning to our simple evidence above, let $\tau_1 < \tau_2 \ldots, \tau_{n-1} < \tau_n$ be all the transition times in $\sigma_{[0,T)}$, $\tau_0 = 0$ and $\tau_{n+1} = T$. The likelihood of drawing σ from the target distribution $P_{\mathcal{N}}$ is

$$\tilde{L}_{\mathcal{N}}(\sigma) = \prod_{X \in \boldsymbol{X}} \prod_{i=0}^{n} \tilde{L}_X(x[\tau_i : \tau_{i+1}))$$

Let $\tilde{L}'_X(x[t_1 : t_2))$ be the corresponding probability density for our sampling procedure. Since we force the values and transitions of variables in $\boldsymbol{V}$ according to the evidence, the probability that we sample an interval $x[\tau_i : \tau_{i+1})$ for $X \in \boldsymbol{V}$ from proposal distribution P' is always 1. Therefore, the likelihood of drawing σ from the proposal distribution P' is

$$\begin{aligned}
\tilde{L}'_{\mathcal{N}}(\sigma) &= \prod_{X \in \boldsymbol{X}} \prod_{i=0}^{n} \tilde{L}'_X(x[\tau_i : \tau_{i+1})) \\
&= \prod_{X \in (\boldsymbol{X} \setminus \boldsymbol{V})} \prod_{i=0}^{n} \tilde{L}_X(x[\tau_i : \tau_{i+1})) \times \prod_{X \in \boldsymbol{V}} \prod_{i=0}^{n} 1.
\end{aligned}$$

To compute the proper weight $w(\sigma)$ we substitute in Equation 6, and get

$$\begin{aligned}
w(\sigma) = \frac{P_{\mathcal{N}}(\sigma, \mathbf{e})}{P'(\sigma)} &= \frac{\prod_{X \in \boldsymbol{X}} \prod_{i=0}^{n} \tilde{L}_X(x[\tau_i : \tau_{i+1}))}{\prod_{X \in (\boldsymbol{X} \setminus \boldsymbol{V})} \prod_{i=0}^{n} \tilde{L}_X(x[\tau_i : \tau_{i+1}))} \\
&= \prod_{X \in \boldsymbol{V}} \prod_{i=0}^{n} \tilde{L}_X(x[\tau_i : \tau_{i+1})).
\end{aligned}$$

Therefore, the weight $w(\sigma)$ is the likelihood contribution of all the variables in $\boldsymbol{V}$. This algorithm exactly corresponds to *likelihood weighting* in Bayesian networks (Shachter and Peot, 1989; Fung and Chang, 1989).

Intuitively, this makes sense because we can account for all the evidence by simply assigning the observed trajectories to the observed variables.

3.5 General Evidence

Now, consider a general evidence pattern $\mathbf{e}$, in which we have time instants where variables become observed or unobserved. How can we force our trajectory to be consistent with $\mathbf{e}$? Suppose there is a set of variables which has evidence beginning at t_e. We can not simply force a transition at time t_e to make the variables consistent with the evidence $\mathbf{e}$: if the set contains more than one variable, the sample would have multiple simultaneous transitions, an event whose likelihood is zero.

Instead, we look ahead for each variable we sample. If the current state of the variable does not agree with the upcoming evidence, we force the next sampled transition time to fall before the time of the conflicting evidence. To do this, we sample from a truncated exponential distribution instead of the full exponential distribution. In particular, if we are currently at time t and there is conflicting evidence for X at time $t_e > t$, we sample from an exponential distribution with the same q value as the normal sampling procedure, but where the sample for Δt (the time to the next transition) is required to be less than $t_e - t$. The probability density of sampling Δt from this truncated exponential is $\frac{q\exp(-q\Delta t)}{1-\exp(-q(t_e-t))}$ where q is the relevant intensity for the current state of X (the diagonal element of $\mathbf{Q}_{X|\mathbf{U}_X}$ corresponding to the current state of X).

The subsequent state is still sampled from the same (forward sampling) distribution. In Section 3.6 we explore a more intelligence option. Note that we cannot, in general, transition directly to the evidence state, as such a transition may not be possible (have 0 rate). Furthermore, if we are still "far away" from the upcoming evidence, such a transition may lead to a highly unlikely trajectory resulting in an inefficient algorithm.

To calculate the weight $w(\sigma)$, we partition σ into two pieces. Let σ_e be the collection for all variables $X \in \boldsymbol{X}$ of intervals $x[t_1 : t_2)$ where the behavior of X is set by the evidence. Let σ_s be the complement of σ_e containing the collection of intervals of unobserved behavior for all variables. By applying Equation 6, we have

$$\begin{aligned} w(\sigma) &= \frac{P_{\mathcal{N}}(\sigma, \mathbf{e})}{P'(\sigma)} \\ &= \prod_{x[\tau_i : \tau_{i+1}) \in \sigma_s} \frac{\tilde{L}_X(x[\tau_i : \tau_{i+1}))}{\tilde{L}'_X(x[\tau_i : \tau_{i+1}))} \times \prod_{x[\tau_i : \tau_{i+1}) \in \sigma_e} \frac{\tilde{L}_X(x[\tau_i : \tau_{i+1}))}{\tilde{L}'_X(x[\tau_i : \tau_{i+1}))} \\ &= \prod_{x[\tau_i : \tau_{i+1}) \in \sigma_s} \frac{\tilde{L}_X(x[\tau_i : \tau_{i+1}))}{\tilde{L}'_X(x[\tau_i : \tau_{i+1}))} \times \prod_{x[\tau_i : \tau_{i+1}) \in \sigma_e} \tilde{L}_X(x[\tau_i : \tau_{i+1})). \end{aligned} \tag{7}$$

Based on the distribution we sampled for transition time of the variable in each step, we can further partition σ_s into three pieces:

σ_{sn} be the collection for all variables $X \in \boldsymbol{X}$ of intervals $x[t_1 : t_2)$ where the transition time is sampled from an exponential distribution.

σ_{st} be the collection for all variables $X \in \boldsymbol{X}$ of intervals $x[t_1 : t_2)$ where the transition time is sampled from a truncated exponential distribution and the variable is involved in the next transition.

σ_{sf} be the collection for all variables $X \in \boldsymbol{X}$ of intervals $x[t_1 : t_2)$ where the transition time is sampled from a truncated exponential distribution and the variable is not involved in the next transition.

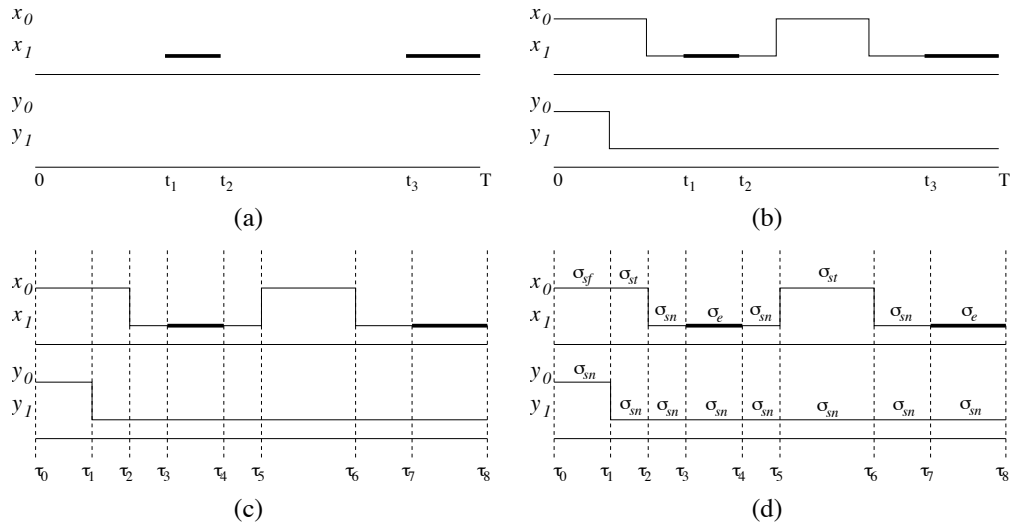


Figure 3: (a) Evidence of a CTBN. (b) A sampled trajectory agreeing with the evidence. (c). Partitioning of the trajectory according to the evidence and the transitions. σ_e equals $x[\tau_3 : \tau_4)$ and $x[\tau_7 : \tau_8)$ (d) Partitioning of the trajectory based on the different sampling situations.

Therefore, we can rewrite Equation 7 as

$$w(\sigma) = \prod_{x[\tau_i : \tau_{i+1}) \in \sigma_{sn}} \frac{\tilde{L}_X(x[\tau_i : \tau_{i+1}))}{\tilde{L}'_X(x[\tau_i : \tau_{i+1}))} \times \prod_{x[\tau_i : \tau_{i+1}) \in \sigma_{st}} \frac{\tilde{L}_X(x[\tau_i : \tau_{i+1}))}{\tilde{L}'_X(x[\tau_i : \tau_{i+1}))} \times \prod_{x[\tau_i : \tau_{i+1}) \in \sigma_{sf}} \frac{\tilde{L}_X(x[\tau_i : \tau_{i+1}))}{\tilde{L}'_X(x[\tau_i : \tau_{i+1}))} \times \prod_{x[\tau_i : \tau_{i+1}) \in \sigma_e} \tilde{L}_X(x[\tau_i : \tau_{i+1})). \quad (8)$$

Example 2 *Assume that we are given a CTBN with two binary variables X and Y. X has two states x_0 and x_1. Y has two states y_0 and y_1. We have such observation: X is x_1 in interval $[t_1, t_2)$ and $[t_3, T)$, as shown in Figure 3(a). To answer queries based on the evidence, we use the method above to sample trajectories. Figure 3(b) shows one of the sampled trajectories. To calculate the weight of the trajectory, we partition the trajectory into four categories (as shown in Figure 3(c) and Figure 3(d)), and apply Equation 8.*

According to Equation 8, each time we add a new transition to the trajectory, we advance time from t to $t + \Delta t$. For each variable x we must update the weight of trajectory to reflect the likelihood ratio for $x[t : t + \Delta t]$ based on the distribution we use to sample the "next time" and the transition variable we select. Each such variable can be considered separately as their times are sampled independently.

For any variable x whose value is given in the evidence during the interval $[t, t + \Delta t)$, as we discussed above, the contribution to the trajectory weight is just $\tilde{L}_{\mathcal{N}}(x[t : t + \Delta t))$. For any variable $x[t : t + \Delta t) \in \sigma_{ns}$, whose "next time" was sampled from an exponential distribution, $\tilde{L}_X(x[\tau_i : \tau_{i+1})) = \tilde{L}'_X(x[\tau_i : \tau_{i+1}))$ and the ratio is 1.

Now, we consider segments $x[t : t + \Delta t) \in \sigma_{st}$ and $x[t : t + \Delta t) \in \sigma_{sf}$. The behavior of the variables in these segments are forced due to upcoming evidence.

For variable X that $x[t : t + \Delta t) \in \sigma_{st}$, the variable's "next time" is sampled from a truncated exponential distribution and it is part of the next transition. The weight must be multiplied by the probability density of sampling the transition in $P_{\mathcal{N}}$ divided by the the probability density in the sampling algorithm. The former is

an exponential distribution and the latter is the same exponential distribution, truncated to be less than $t_e - t$. The ratio of these two probabilities is $1 - \exp(-q(t_e - t))$, where q is the relevant intensity.

Otherwise, $x[t : t + \Delta t) \in \sigma_{sf}$, the next time for the variable was sampled from a truncated exponential but was longer than Δt. In this case, the ratio of the probabilities of a sample being greater than Δt is $\frac{1-\exp(-q(t_e-t))}{1-\exp(-q(t_e-t-\Delta t))}$. Note that when Δt is small (relative to $t_e - t$, the time to the next evidence point for this variable), the ratio is almost 1. So, while the trajectory's weight is multiplied by this ratio for every transition for every variable that does not agree with the evidence, it does not overly reduce the weight of the entire trajectory.

The algorithm for CTBN importance sampling is shown in Figure 4. To more easily describe the evidence, we define a few helper functions:

$\mathbf{e}_X^{val}(t)$ is the value of X at time t according to the evidence, or undefined if X has no evidence at t.

$\mathbf{e}_X^{time}(t)$ is the first time after t when $\mathbf{e}_X^{val}(t)$ is defined.

$\mathbf{e}_X^{end}(t)$ is the first time after or equal to t when $\mathbf{e}_X^{val}(t)$ changes value or becomes undefined.

Note that $\mathbf{e}_X^{end}(t) = t$ when there is point evidence at t, when t is the end of an interval of evidence, and when there is a transition in the evidence at time t.

The line numbers follow those given in the forward sampling algorithm with new or changed lines marked with an asterisk. $Time(X)$ might be set to the end of an interval of evidence which is not a transition time but simply a time when we need to resample a next potential transition. This means that we will not update σ with a new transition every time through the loop. The algorithm differs from the forward sampling procedure as follows. Step 2 now accounts for evidence at the beginning of the trajectory (using standard likelihood weighting for Bayesian networks). In Step 3, we draw Δt from the truncated exponential if the current value disagrees with upcoming evidence. If the current evidence includes this variable, Δt is set to the duration of such evidence. Step 5 updates the weights using the procedure *Update-Weight*. Finally, Step 7 now deals with variables that are just leaving the evidence set.

3.6 Predictive Lookahead

The algorithm in Figure 4 draws the next state for a variable from the same distribution as the forward sampling algorithm. This may cause a variable to transition several times in a short interval before evidence as the variable "searches" to find a way to transition into the evidence. Thus, we may generate many unlikely samples, making the algorithm inefficient. We can help mitigate this problem by trying to force the variable into a state that will lead to the evidence.

When sampling the next state for variable X at time t, instead of sampling from the multinomial according to $\boldsymbol{\theta}_{x(t)|\mathbf{u}_X(t)}$, we would like to sample from the distribution of the next state conditioned on the upcoming evidence. Suppose X is in state x_i at time t, and the next evidence for X is state x_k at t_e. Assuming the parents of X do not change before t_e and ignoring evidence over the children of X, the distribution of the state of X at t given only the evidence can be calculated using Equation 3:

$$\tilde{P}(X_{t+\Delta t} = x_j | X[t : t + \Delta t) = x_i, X_{t_e} = x_k) = \frac{1}{Z}\mathbf{1}_j^\top \mathbf{Q}_X \exp(\mathbf{Q}_X(t_e - t))\mathbf{1}_k = p_{i,j}$$

where $\mathbf{1}_j$ is the vector of zeros, except for a one in position j. We can therefore select our new state according to the distribution of $\tilde{P}(X_{t+\Delta t} | X[t : t + \Delta t) = x_i, X_{t_e} = x_k)$ and, assuming state x_j is selected, multiply the weight by $\frac{\theta_{x_i x_j|\mathbf{u}_X(t)}}{p_{i,j}}$ to account for the difference between the target and sampling distributions.

3.7 Particle Filtering

The algorithm in Figure 4 allows us to generate a single trajectory and its weight, given the evidence. To apply this algorithm to the task of online inference in a dynamic system, we can generate multiple trajectories in parallel, advancing time forward as evidence is obtained.

Procedure *CTBN-Importance-Sample*$(t_{end}, \mathbf{e})$

1. $t \leftarrow 0, \sigma \leftarrow \emptyset, w \leftarrow 1$ *
2. For each variable $X \in \boldsymbol{X}$
   If $\mathbf{e}_X^{val}(0)$ defined,
   set $x(0) \leftarrow \mathbf{e}_X^{val}(0)$, *
   Set $w \leftarrow w \cdot \theta^{\mathcal{B}}_{x(0)|\mathbf{pa}_{\mathcal{B}}(0)}$ *
   Else
   choose state $x(0)$ according to $\boldsymbol{\theta}^{\mathcal{B}}_{X|\mathbf{pa}_{\mathcal{B}}(X)}$

Loop:

3. For each $X \in \boldsymbol{X}$ such that $Time(X)$ is undefined:
   If $\mathbf{e}_X^{val}(t)$ is defined, set $\Delta t \leftarrow \mathbf{e}_X^{end}(t) - t$ *
   Elseif $\mathbf{e}_X^{val}(t_e)$ is defined where $t_e = \mathbf{e}_X^{time}(t), x(t) \neq \mathbf{e}_X^{val}(t_e)$,
   choose Δt from an exponential distribution with
   parameter $q_{x(t)|\mathbf{u}_X(t)}$ given $\Delta t < (t_e - t)$. *
   Else choose Δt from an exponential w/ param. $q_{x(t)|\mathbf{u}_X(t)}$
   Define $Time(X) \leftarrow t + \Delta t$
4. Let $X = \arg\min_{X \in \boldsymbol{X}} [Time(X)]$
5. If $Time(X) \geq t_{end}$
   $w \leftarrow$ *Update-Weight*(X, w, t, t_{end}) *
   return (σ, w)
   Else *
   $w \leftarrow$ *Update-Weight*$(X, w, t, Time(X))$ *
6. Update $t \leftarrow Time(X)$
7. If $\mathbf{e}_X^{end}(t) \neq t$ or $\mathbf{e}_X^{val}(t)$ is defined *
   If $\mathbf{e}_X^{val}(t)$ is defined, set $x(t) \leftarrow \mathbf{e}_X^{val}(t)$ *
   Else choose $x(t)$, the next value of X, from a
   multinomial with parameter $\boldsymbol{\theta}_{x(t)|\mathbf{u}_X(t)}$
   Add $\langle X \leftarrow x(t), t \rangle$ to σ.
   Undefine $Time(X)$ and $Time(Y)$ for all variables Y
   for which $X \in \mathbf{U}_Y$
   Else *
   Undefine $Time(X)$. *

Procedure *Update-Weight*(Y, w, t_1, t_2)

1. For each $X \in \boldsymbol{X}$ such that $\mathbf{e}_X^{val}(t)$ is defined for $t \in [t_1, t_2)$:
   $w \leftarrow w \cdot \tilde{L}_X(x[t_1 : t_2))$
2. For each $X \in \boldsymbol{X}$ such that $\mathbf{e}_X^{val}(t_e)$ is defined,
   where $t_e = \mathbf{e}_X^{time}(t_1)$, and $x(t_1) \neq \mathbf{e}_X^{val}(t_e)$:
   If $X = Y$, $w \leftarrow w \cdot (1 - \exp(-q_{x(t_1)|\mathbf{u}_X(t_1)}(t_e - t_1)))$
   Else $w \leftarrow w \cdot \frac{1 - \exp(-q_{x(t_1)|\mathbf{u}_X(t_1)}(t_e - t_1))}{1 - \exp(-q_{x(t_1)|\mathbf{u}_X(t_1)}(t_e - t_2))}$
3. **return** w

Figure 4: Importance sampling for CTBNs. Changes from Figure 2 are noted with asterisks.

The resulting algorithm is an instance of sequential importance sampling, and therefore suffers from its characteristic flaw: As the trajectory length increases, the distribution of the importance weights gets increasingly skewed, with most importance weights converging to zero exponentially quickly. Thus, the number of "relevant" samples gets increasingly small, and the estimates provided by the set of samples quickly become meaningless. A family of methods, commonly known as sequential Monte Carlo or particle filtering (Doucet et al., 2001), have been proposed in the setting of discrete-time processes to address this flaw.

Procedure *CTBN-Particle-Filtering*$(\{X_0^i, w_0^i\}_{i=1\ldots N}, t_{end}, \mathbf{e})$
1. $k \leftarrow 0, W_t \leftarrow 1, N_r \leftarrow N$
2. For $i \leftarrow 1$ to N: $Pa_0^i \leftarrow i, w^i \leftarrow 1/N$

Loop:

3. For each i such that $t_k^i < t_{end}$:
   $(X_{k+1}^i, t_{k+1}^i, w_{k+1}^i) \leftarrow$
   *S*ample-Segment$(X_k^{Pa_k^i}, t_k^{Pa_k^i}, w^i, t_{end}, \mathbf{e})$
   If $t_{k+1}^i \geq t_{end}$
   $N_{remain} \leftarrow N_r - 1$,
   $W_t \leftarrow W_t - w_{k+1}^i$
4. $k \leftarrow k+1$
5. If $N_r = 0$
   return $\{X_{m_i}^i, t_{m_i}^i, w_{m_i}^i, Pa_{m_i}^i\}_{i=1\ldots N, m_i=1\ldots n_i}$,
   where n_i is the number of transitions of the i^{th} particle
6. Calculate $\widehat{N_{eff}}$ of all incomplete particles
7. If $\widehat{N_{eff}} < N_{thr}$
   Sample Pa_k^i according to w_k^i
   $w^i \leftarrow W_t \times 1/N_r$
   Else
   $w^i \leftarrow w_k^i, Pa_k^i \leftarrow Pa_{k-1}^i$

Figure 5: Particle Filtering for CTBNs

At a high level, these methods re-apportion our samples to focus more efforts on more relevant samples—those with higher weights.

The application of this idea to our setting introduces some subtleties because different samples are not generally synchronized. We could pick a time t and run the algorithm in Figure 4 with $t_{end} = t$ so that samples are synchronized at t. We would re-apportion the weights and continue each trajectory from its state at t, first setting $Time(X)$ to be undefined for all X. However, choosing the proper synchronization time t is a non-trivial problem which may depend on the evidence and the speed the system evolves.

Instead of synchronizing all the particles by the time, we can align particles by the number of transitions. If we let t_i be the i^{th} transition time and X_i be the value of X from t_{i-1} to t_i, the following recursion holds.

$$\begin{aligned} P(X[0:t_n)) &= P(X_{1:n}, t_{1:n}, e_{[0:t_n)}) \\ &= P(X_{1:n-1}, t_{1:n-1}, e_{[0:t_{n-1})}) P(X_n | X_{n-1}) P(X_{[t_{n-1},t_n)}, e_{[t_{n-1},t_n)} | X_{n-1}, e_{t_{n-1}}). \end{aligned}$$

The weighted approximation of this probability is given by

$$P(X[0:t_n)) \approx \sum_{i=1}^{N} w(X^i[0:t_n)) \delta(X[0:t_n), X^i[0:t_n))$$

where $X^i[0:t_n)$ is the i^{th} sample and $w(X^i[0:t_n))$ is the normalized weight of the i^{th}sample. According to Equation 8, the weight can be updated after every transition step. The weight update equation can be shown as

$$w(X^i[0:t_n)) \propto w(X^i[0:t_{n-1})) \frac{\tilde{L}_X(X^i[t_{n-1}:t_n))}{\tilde{L}'_X(X^i[t_{n-1}:t_n))}.$$

Thus, to sample multiple trajectories in parallel, we apply the CTBN importance sampling algorithm to each trajectory until a transition occurs. To avoid the degeneracy of the weights, we resample the particles when the estimated effective sample size $\widehat{N_{eff}} = \frac{1}{\sum_i (w_k^i)^2}$ is below a threshold N_{thr}. This procedure is similar to the regular particle filtering algorithm except that all particles are not synchronized by time but the number

Procedure *CTBN-Particle-Smoothing*$(\{X^i_{m_i}, t^i_{m_i}, w^i_{m_i}\}, t_{end}, \mathbf{e})$
$i = 1 \ldots N, m_i = 1 \ldots M_i$
1. $\sigma \leftarrow \emptyset$
2. Choose k with probability proportional to $w_i^{M_i}$
3. set $Y = X_{M_k^k}, s \leftarrow M_k, t \leftarrow t_s^k$
Loop:
4. $\sigma_{[t_{s-1},s)} \leftarrow Y$
5. If σ is complete
return σ
6. For $j \leftarrow 1$ to N
$w'_j \leftarrow$ *Check-Weight*$(Y, t, X^j_{s-1}, t^j_{s-1}, w^j_{s-1})$
7. Choose i with probability proportional to w'_i
8. $Y \leftarrow X_i^s, t \leftarrow t_s^j$ $s \leftarrow s-1$

Procedure *Check-Weight*(X, t, X_s, t_s, w_s)
1. If $t \leq t_s$ or $\mathbf{e}_{(ts,t)}$ contains a transition, or
the value of X and X_s do not differ by only one variable
return 0
2. $\sigma_{[t_s,t)} \leftarrow X_s, \sigma(t) \leftarrow X$
3. $w \leftarrow w_s \cdot \tilde{L}_X(\sigma_{[t_s,t_2]})$
4. **return** w

Figure 6: Particle Smoothing for CTBNs

of transitions. To answer queries in the time interval $[0, T)$, we propagate the particles until all of their last transitions are greater than T.

Figure 5 shows the algorithm for generating N trajectories from 0 to T in a CTBN. It assumes that the initial values and the weights have already been sampled. The procedure *Sample-Segment* loops from line 3 to 7 in Figure 4 until a transition occurs, returns the transition time and variables value, and updates the corresponding weight for that segment. Note that we are approximating the distribution $P(X_{1:n}, t_{1:n}, e_{[0:t_n)})$ for all possible n. Therefore, we only propagate and re-apportion weights for particles that have not yet reached time T. Particles that have been sampled past T are left untouched.

3.8 Particle Smoothing

Although the resampling step in the particle filtering algorithm reduces the skew of the weights, it leads to another problem: the diversity of the trajectories is also reduced since particles with higher weights are likely to be duplicated multiple times in the resampling step. Many trajectories share the same ancestor after the filtering procedure. A Monte Carlo smoothing algorithm using backward simulation addresses this problem (Godsill et al., 2004).

The smoothing algorithm for discrete-time systems generates trajectories using N weighted particles $\{x^i_t, w^i_t\}$ from the particle filtering algorithm. It starts with the particles at time T, moves backward one step each iteration and samples a particle according to the product of its weight and the probability of it transitioning to the previously sampled particle. Specifically, in the first step, it samples $\widetilde{x}_T$ from particles x^i_T at time T with probability w^i_T. In the backward smoothing steps it samples $\widetilde{x}_t$ according to $w^i_{t|t+1} = w^i_t f(\widetilde{x}_{t+1}|x^i_t)$, where $f(\widetilde{x}_{t+1}|x^i_t)$ is the probability that the particle transitions from state x^i_t to $\widetilde{x}_{t+1}$. The resulting trajectory set is an approximation of $P(x_{1:T}|y_{1:T})$ where $y_{1:T}$ is the observation.

This idea can be used in our setting with modification. Given the filtered particles $\{X^i_{m_i}, t^i_{m_i}, w^i_{m_i}\}$,we need to sample both variable values and transition times at each step when we move backward. There are two main differences from the algorithm in Godsill et al. (2004): There are fewer than N particles that can be

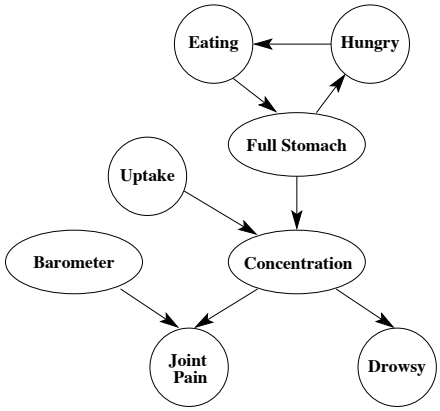


Figure 7: Drug Effect Network

used at the beginning steps of the backward smoothing since the trajectories do not have exactly the same number of transitions, and not all particles at step n can be considered as candidates to move backward. A particle $\{X_n^i, t_n^i, w_n^i\}$ is a valid candidate as the predecessor for $\{\widetilde{X}_{n+1}, \widetilde{t}_{n+1}\}$ only if (1) $t_n^i < \widetilde{t}_{n+1}$, (2) the values of X_n^i and $\widetilde{X}_{n+1}$ differ in only one variable (thus a single transition is possible), and (3) $\mathbf{e}_{(t_n^i, \widetilde{t}_{n+1})}$ contains no transitions.

Figure 6 shows the smoothing algorithm which generates a trajectory from the filtering particles. We apply the algorithm N times to sample N trajectories. These equally weighted trajectories can be used to approximate the smoothing distribution $P(X_{[0,T)}|\mathbf{e})$. Generating one trajectory with this smoothing process requires considering all the particles at each step. The running time of sampling N trajectories using particle smoothing is N times of that of particle filtering.

4. Experimental Results

In this section, we report on the performance of our algorithm on synthetic networks and a network built from a real data set of people's life histories. We tested our algorithm's accuracy for the task of inference and parameter estimation. We also compare our algorithms with other approximate inference algorithms for CTBNs: the method based on the expectation propagation in Saria et al. (2007) and the method based on Gibbs sampling in El-Hay et al. (2008).

All the algorithms we used in the experiments were implemented in the same code base to make fair comparisons. We tried our best to optimize all the code. The implementations are general so that they can be applied to any CTBN model. Our implementation of EP is adapted from that of Saria et al. (2007) who were kind enough to share their code. The code base is described in Shelton et al. (2010) and is available from the authors' website.

4.1 Networks

In our experiments, different types of network structures were used, including the drug effect network (Nodelman et al., 2002), a chain-structured network, and the BHPS network (Nodelman et al., 2005b). All the networks are at the upper size limit for the exact inference algorithm so that we can compare our result to the true value.

Drug Effect Network: The drug effect network is a toy model of the effect of a pain-relief medicine. It has 8 (5 binary and 3 ternary) variables. The structure of the network is shown in Figure 7. At $t = 0$ the person is not hungry, is not eating, has an empty stomach and is not drowsy. He has joint pain due to the falling barometric pressure and takes the drug to alleviate the pain.

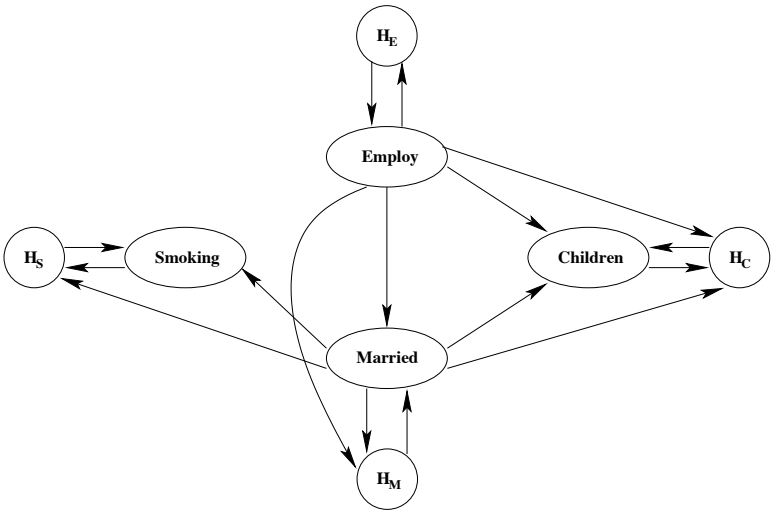


Figure 8: British Household Panel Survey Network

Chain Structured Network: The chain network contains five nodes $X_0, \ldots, X_5$, where X_i is the parent of X_{i+1} for $i < 5$. Each node has five states, $s_0, \ldots, s_4$. X_0 (usually) cycles in two loops: $s_0 \to s_1 \to s_3 \to s_0$ and $s_0 \to s_2 \to s_4 \to s_0$. All the other nodes stay at their current state if it matches their parent and otherwise transition to their parent's state with a high probability. Each variable starts in state s_0.

More specifically, the intensity matrix of X_0 is

$$\mathbf{Q}_{X_0} = \begin{bmatrix} -2.02 & 1 & 1 & 0.01 & 0.01 \\ 0.01 & -2.03 & 0.01 & 2 & 0.01 \\ 0.01 & 0.01 & -2.03 & 0.01 & 2 \\ 2 & 0.01 & 0.01 & -2.03 & 0.01 \\ 2 & 0.01 & 0.01 & 0.01 & -2.03 \end{bmatrix}.$$

For all other nodes, the off-diagonal elements of the intensity matrices are given by

$$q_{s_i,s_j|\mathbf{u}=s_k} = \begin{cases} 0.1 & \text{if } i \neq j \text{ and } j \neq k, \\ 10 & \text{if } i \neq j \text{ and } j = k. \end{cases}$$

BHPS Network: This network was learned from the British Household Panel Survey (BHPS) (ESRC Research Centre on Micro-social Change, 2003) data set. The data set provides information about British citizens. The data are collected yearly by asking thousands of households questions such as household organization, employment, income, wealth and health. Similar to Nodelman et al. (2005b), we keep a small set of variables so that exact inference could be applied. We chose four variables: employ (ternary: student, employed, unemployed), children (ternary: 0, 1, 2+), married (binary: not married, married), and smoking (binary: non-smoker, smoker), and we assumed there is a hidden variable (binary) for each of those four variables. We trained the network on 8935 trajectories of people's life histories. We applied the structural EM algorithm in Nodelman et al. (2005b) and learned the structure of the network shown in Figure 8. We then estimated the parameters of the network using the EM algorithm and exact inference. We consider the learned model as the true BHPS network model for these experiments.

4.2 Evaluation Method

We evaluated the performance of the approximate inference algorithms in two tasks: the inference task of answering queries given evidence and the learning task of parametric learning with partially observed data.

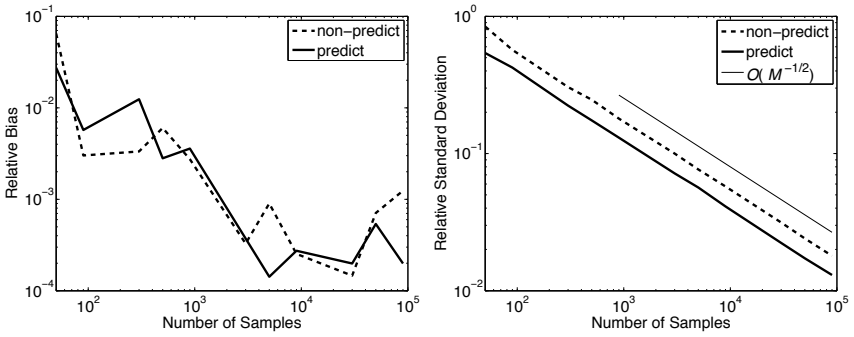


Figure 9: Relative bias and standard deviation of sampling with and without predictive lookahead.

In the inference task, each evidence is a partially observed trajectory of the CTBN network. The evidence is generated using two methods. The first method is to set it manually. The second is to generate a trajectory using the forward sampling algorithm and randomly remove some parts of the sampled trajectory. In particular, we repeated the following procedure n times: for each variable, we randomly removed the information of the trajectory from t_s to $t_s + \gamma T$, where T is the total length of the trajectory, t_s is randomly sampled from the $[0, T - \gamma T]$ uniform distribution and $\gamma < 1$. After we run the removing procedure n times, there are at most $n\gamma$ time units of information missing for each variable. In all comparisons, this procedure was applied once and the same evidence was given to all algorithms.

In our experiments, we set our query to be one of three types: the expected total amount of time a variable X stays on some state x_i, the expected total number of times that a variable transitions from state x_i to state x_j, or the distribution of variable at time t.

For each query, we ran the sampling based algorithms with different sample sizes, M. For each sample size, we ran the experiment N times. We calculated our query according to Equation 6 and compared the result to the true value calculated using exact inference. We used two metrics: the relative bias $\frac{|\sum v_M - v^*|}{v^* N}$, where v_M is the query value of sampling algorithm with sample size M, and v^* is the true value; and the relative standard deviation $\frac{\sigma_M}{v^*}$ where σ_M is the standard deviation from the true value when sample size is M. For each sample size, we also recorded the average running time $\bar{t}_M$ of each experiment and used $\bar{t}_M$ to evaluate the efficiency of the algorithm.

In the learning task, we used the sampling algorithms to estimate the parameters of a CTBN network given some partially observed data. Monte Carlo EM (Wei and Tanner, 1990) was applied in this task: In each iteration, we used the sampling based algorithm to estimate the expected sufficient statistics given the incomplete data and used Equation 4 to compute the parameters.

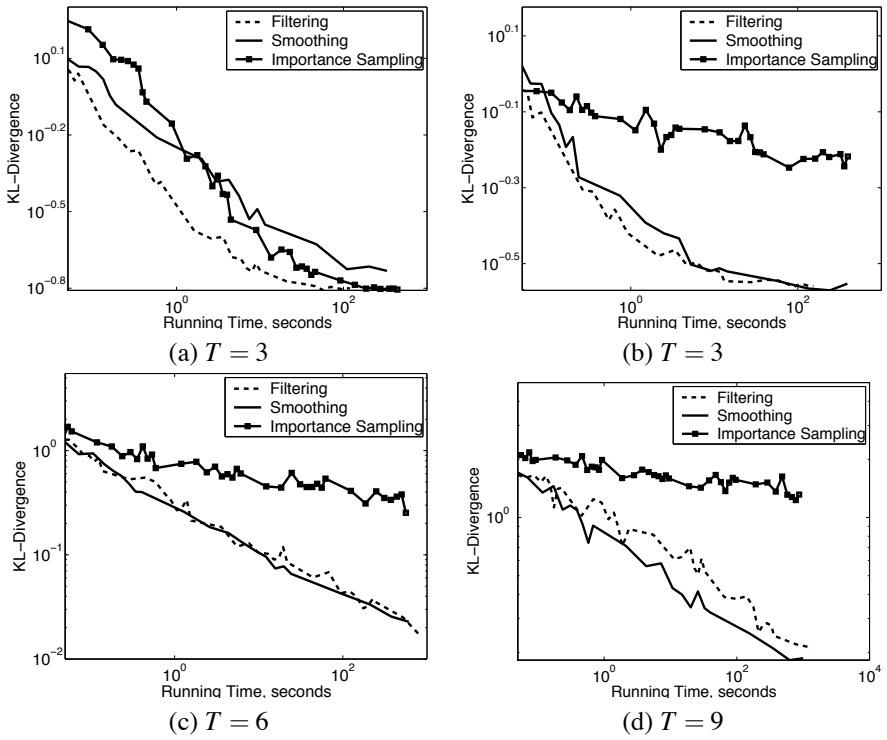
The training data were generated by sampling trajectories from the true model and randomly removing some portion of the information as described above. We sampled another set of trajectories from the true model as the testing data. We calculated the log-likelihood of the testing data under the learned model to evaluate the learning accuracy.

4.3 Inference Experimental Results

In this section, we evaluate the performance of our importance sampling based algorithms in answering queries and compare with the EP algorithm in Saria et al. (2007) and the Gibbs sampling algorithm in El-Hay et al. (2008).

4.3.1 COMPARISON OF IMPORTANCE SAMPLING AND PREDICTIVE LOOKAHEAD

We first tested the importance sampling algorithm and the predictive lookahead modification using the drug effect network. We set the observed evidence: on $t = [0,1)$ the stomach is empty, on $t = [0.5, 1.2)$ the



(a) $T = 3$ (b) $T = 3$

(c) $T = 6$ (d) $T = 9$

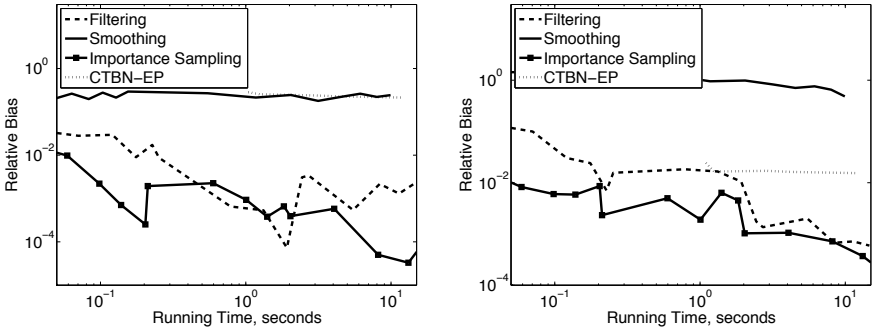
Figure 10: Time-efficiency comparison of particle filtering, smoothing and importance sampling

barometer is falling, and on $t = [1.5, 2.5)$ he is drowsy. Our query is the expected total amount of time that he has no joint pain on $[0, 2.5)$. (The true value is 0.1093). We ran the two algorithms with sample sizes, M, from 5 to 90000. For each sample size, we ran the algorithms $N = 1000$ times. The results are shown in Figure 9. Both algorithms achieve the correct result when the sample size is large. The standard deviation decreases at a rate of $O(\frac{1}{\sqrt{M}})$ (shown by the thin solid line). The sampling algorithm with prediction achieves lower standard deviation than the non-prediction version.

4.3.2 IMPORTANCE SAMPLING, PARTICLE FILTERING AND SMOOTHING

We then used the chain network to evaluate the efficiency of the importance sampling, particle filtering, and smoothing algorithms. We assumed that only X_4 was observed in this experiment. We used four different evidences. The first one is a simple evidence: only part of the behavior of X_4 is observed: on $[1, 1.7)$, $X_4 = s_3$, and on $[2, 2.5)$, $X_4 = s_2$. For the other three, the behavior of X_4 is fully observed during the interval $[0, T)$, where $T = 3, 6, 9$. This is done by forward sampling a trajectory from 0 to T and keeping only the information about X_4. Our query is the marginal distribution $P(X_2(\frac{T}{2})|\mathbf{e}_{[0,T)})$. Note that this is the most difficult case for the importance sampling algorithm since the chain network is nearly deterministic. We recorded the average running time and KL-divergence between the estimated and true distributions, for each sample size across $N = 300$ trials.

Figure 10 shows the efficiency of the three algorithms. In Figure 10(a), we used the simple evidence. In Figure 10 (b)-(d), we used the evidence with X_4 fully observed and $T = 3, 6, 9$ respectively. In all four cases, the particle filtering and smoothing algorithms both outperform the importance sampling algorithm when the sample size is small (small running time). For simple evidence (Figure 10(a)), the importance sampling algorithm achieves comparable performance when the sample size is large. When the evidence is complicated (Figure 10 (b)-(d)), the error of importance sampling is large, even when we use very large



(a) Concentration (b) Joint pain

Figure 11: Comparison to expectation propagation: Drug Network

sample sizes. When the trajectory is short, the particle filtering algorithm is slightly better than the particle smoothing algorithm. This is because the filtering algorithm can generate more samples than the smoothing algorithm with the same running time. However, as the trajectory length increases, the particle smoothing algorithm outperforms the filtering algorithm due to particle diversity problems.

4.3.3 COMPARISON OF IMPORTANCE SAMPLING AND EP

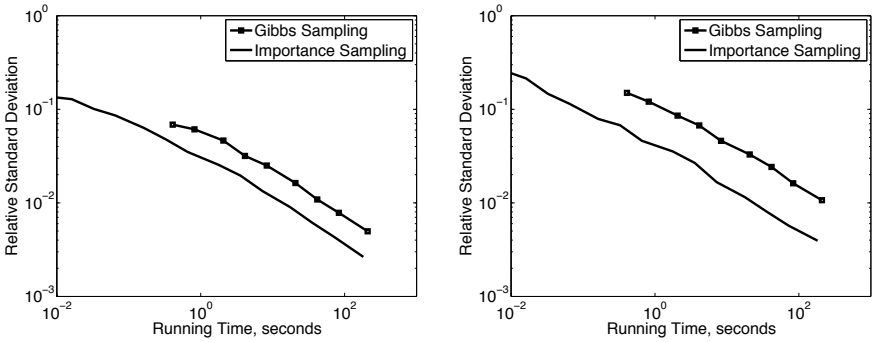
We also compared our three sampling algorithms to the approximate inference algorithm based on expectation propagation in Saria et al. (2007). We did not use their adaptive splitting method (for reasons we explain below). Even without the adaptive splitting, their method still differs from that of Nodelman et al. (2005a), in that it allows asynchronous propagation of messages along time.

We used the same evidence as in Section 4.3.1 on the drug effect network and answered two queries: the total amount of time that the concentration is low and the total amount of time the person has no joint pain. For the EP algorithm, we first tried a segmentation that split the timeline at the evidence. We then gradually decreased the time interval of the segments to 0.15. The results of accuracy with respect to running time are shown in Figure 11. The importance sampling algorithm and the particle filtering algorithm outperforms the EP algorithm in answering both queries. Among the sampling-based algorithms, the importance sampling algorithm performs the best and the smoothing algorithm is the worst. This is not surprising given that most of the nodes are binary. At each transition time, the sampled trajectory has no choice as to the next state. Therefore, smoothing (or filtering) has less effect as there is no need to intelligently select the next state. However, the extra computation time for resampling and backward simulation makes the filtering and smoothing algorithm less efficient.

As mentioned above, we did not employ the adaptive splitting method of Saria et al. (2007). It would not have changed our results much. The left-most points in our Figure 11 correspond to the minimum number of splits. (They are as fast as possible.) The right-most points of the Figure 11 correspond to many fine splits, and are about as accurate as possible, and we can see that the accuracy has flattened out. So, while the horizontal widths of the EP curves would have been shortened (by allowing for the better accuracy in less time), the vertical spread would have been approximately the same. In neither plot of Figure 11 would this have made a large difference in the comparisons to our sampling method.

4.3.4 COMPARISON OF IMPORTANCE SAMPLING AND GIBBS SAMPLING

We compared our importance sampling algorithm to the Gibbs Sampling algorithm in El-Hay et al. (2008). We used three CTBN network models: the drug effect network, the BHPS network and the chain structured network. For each network, we randomly generated evidence using the procedure described in Section 4.2.



(a) Query of Time (b) Query of Transition

Figure 12: Comparison to Gibbs Sampling: Drug Network. Note burn-in time for Gibbs Sampling is not included (3.94 seconds on average).

We ran the procedure 4 times for each variable and each time, we removed 20% of the content. Thus, there are at most 80% information missing for each variable.

For the importance sampling algorithm, we chose the sample size M from 10 to 500000. For Gibbs sampling algorithm, we chose the sample size M from 10 to 5000. We ran the experiments for each sample size $N = 100$ times and recorded the average running time for each algorithm. For Gibbs sampling algorithm, we first ran 100 "burn-in" iterations for each sample size before we sample trajectories from the sampler. The time spent on the "burn-in" iterations was not included in the final running time.

For the drug effect network, the evidence trajectory begins at time $t = 0$ and ends at time $t = 5$. We asked two queries: the expected total amount of time the person's stomach is half full, and the expected number of times that the person's stomach changes from empty to half full.

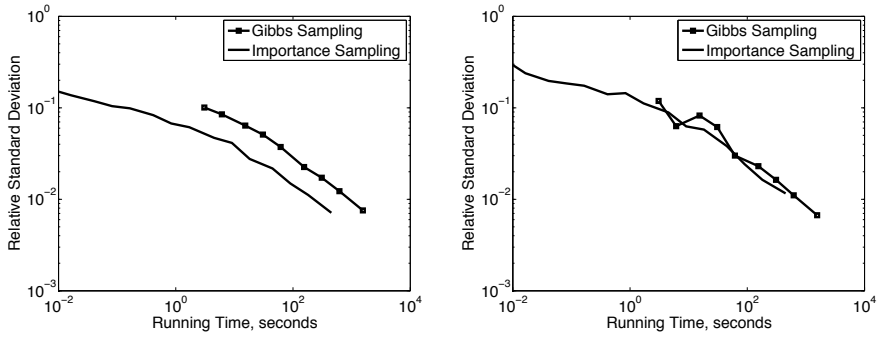
Using enough running time (sample size), we observed that both algorithms could answer the queries accurately (with a relative bias below 0.1%). The decreasing of the relative standard deviation with respect to the running time of the two algorithms are shown in Figure 12. The average "burn-in" time for Gibbs sampler is about 3.94 seconds. From the figure, we can see that importance sampling outperforms Gibbs sampling in answering both queries.

For the BHPS network, we set the evidence from $t = 0$ to $t = 50$ (years). We asked similar queries: the expected total amount of time a person's employment status is as a student and the expected number of times that he becomes employed. We chose the same sample sizes as on the drug effect network and ran each sample size $N = 100$ times. Figure 13 shows the result of the decreasing of the standard deviation of the two algorithms. The average "burn-in" time for Gibbs sampling algorithm in this experiment is 30.88 seconds.

We achieved similar result as the experiments with the drug effect network. The importance sampling algorithm outperformed the Gibbs sampling algorithm in answering the query of time. The performances on the query of transitions are almost the same.

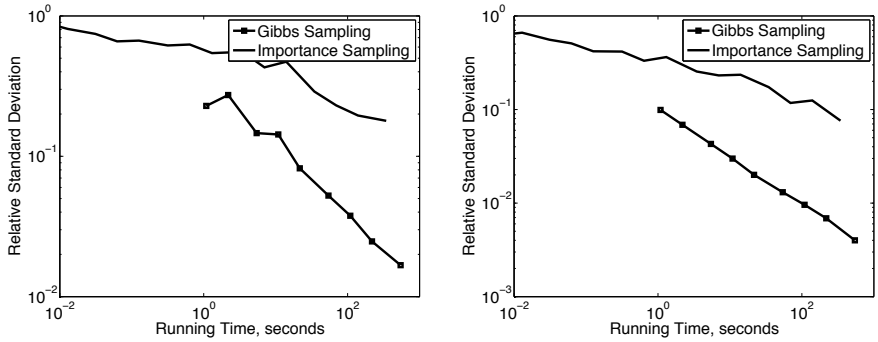
In both networks, importance sampling outperformed Gibbs sampling in three of the four cases, even when the running time on "burn-in" iterations was not considered. To achieve the same accuracy and standard deviation, Gibbs sampling algorithm requires fewer samples. This is because for each variable, Gibbs sampling samples from the true posterior distribution given the evidence and its Markov blanket. However, sampling from the true posterior distribution is computational costly, since it requires repeatedly computing the conditional cumulative distribution function. Using the same amount of time, importance sampling can sample far more trajectories, which outperforms Gibbs sampling.

We last compared these two algorithms using the chain structured network. The evidence trajectory begins at time $t = 0$ and ends at time $t = 5$. We set the queries to be the expected total amount of time X_2 stays in



(a) Query of Time (b) Query of Transition

Figure 13: Comparison to Gibbs Sampling: BHPS Network. Note burn-in time for Gibbs Sampling is not included (30.88 seconds on average).



(a) Query of Time (b) Query of Transition

Figure 14: Comparison to Gibbs Sampling: Chain Network. Note burn-in time for Gibbs Sampling is not included (11.42 seconds on average).

state s_1 and the expected number of times that X_2 transitions from s_0 to s_1. Figure 14 shows the result over $N = 100$ runs. The average "burn-in" time for Gibbs sampling algorithm in this experiment is 11.42 seconds.

The Gibbs sampling algorithm achieved better performance in this experiment. The result is not surprising. As we have mentioned before, the chain structured network is nearly deterministic, and it is the hardest case for the importance sampling algorithm. We further examined the randomly generated evidence. The only observed state on X_0 is s_0, which makes this experiment even harder for the importance sampling algorithm. However, it is a very easy case for the Gibbs sampling algorithm since it is nearly deterministic and is structurally simple. (There are only at most one parent and one child for each node.) Although importance sampling can generate many more samples in the same period of time, most of these samples are trajectories with very small weights.

4.4 Parameter Estimation Experimental Results

In this section, we evaluate the performance of importance sampling algorithm on parameter estimation and compare to the Gibbs sampling algorithm and the EP algorithm.

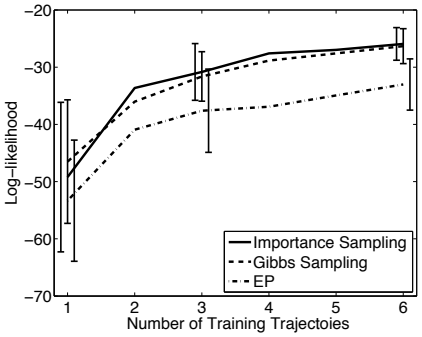


Figure 15: Learning results for the drug effect network. Standard deviations across training data selection (and random samples for the sampling methods) are shown jittered slightly for clarity.

We used the drug effect network for this experiment. We sampled increasing We sampled increasing numbers of trajectories of 5 time lengths. To hide part of the trajectory, we did the following: In each iteration, for each variable we randomly selected a time window of 0.5 time lengths and removed the content in that window. We repeated this until we dropped 50% of the content of the trajectory. We used these incomplete trajectories as our training data. We sampled another 50 trajectories with the same length as our testing data.

To estimate the parameters of the CTBN network, we followed the EM algorithm in Nodelman et al. (2005b). When calculating the expected sufficient statistics, importance sampling, Gibbs sampling, and expectation propagation were used. Therefore, the likelihood in the E-step was calculated approximately. In our experiment, we fixed the total number of iterations for the EM algorithm at 15. In each iteration, we compared the calculated likelihood to the likelihood in the previous iteration. If the likelihood decreased, we kept the parameters in the previous iteration.

We chose the initial parameters for the EM algorithm by sampling the diagonal elements of the conditional intensity matrices from the Gamma distribution with parameters $(0.5, 1)$ and sampling the transition probabilities from a Dirichlet distribution. We randomly sampled 5 models as the initial parameters for the EM algorithm. For each initial parameter set, we ran the EM algorithm 10 times for the sampling methods (and once for EP which is deterministic). We evaluated the learning accuracy by calculating the average log-likelihood of the testing data on the 50 learned networks. To compare the running efficiency of the two sampling-based algorithms, we fixed the total amount of time for the sampler to generate samples in each EM iteration to be the same time as the EP algorithm took (approximately 23 seconds). For the Gibbs sampling algorithm, we dropped the first 50 trajectories as "burn-in" iterations. Figure 15 shows the results as we increased the number of training trajectories from 1 to 6.

All algorithms obtain higher log-likelihood on the testing data when we increase the number of training trajectories. The sampling methods do better (and have lower variation), especially as the data size grows.

5. Conclusion

We have presented an approximate inference algorithm with two variations based on importance sampling. We naturally extended the algorithm to sequential Monte Carlo methods such as particle filtering and smoothing in CTBNs. We applied our sampling algorithm to synthetic networks and a network from real data. We evaluated the efficiency of our algorithms and compared to other approximate inference algorithms based on expectation propagation and Gibbs sampling. Our importance sampling algorithm outperformed both in most of the experiments presented in this paper. In the situation of a highly deterministic system, Gibbs sampling performed better.

The networks used in this paper are at the upper size limit for exact computation. For example, calculating the expected sufficient statistics of the chain structured network given evidence takes more than two days using exact inference. Thus, approximate inference methods are critical for tracking, predicting, and learning in continuous time Bayesian networks for real applications. Our importance sampling based algorithms are fast, simple to implement and can be used to calculate the expected value of any function of a trajectory, including the expected sufficient statistics necessary for expectation-maximization for parameter estimation with missing data.

Acknowledgments

We would like to thank Suchi Saria for sharing her EP code and Tal El-Hay for sharing his Gibbs sampling code. This work was funded by AFOSR (FA9550-07-1-0076) and DARPA (HR0011-09-1-0030).

References

Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, Inc., 1998.

Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag Telos, 2001.

Tal El-Hay, Nir Friedman, and Raz Kupferman. Gibbs sampling in factorized continuous-time Markov processes. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, 2008.

ESRC Research Centre on Micro-social Change. British household panel survey. Computer Data File and Associated Documentation. http://www.iser.essex.ac.uk/bhps, 2003. Colchester: The Data Archive.

Yu Fan and Christian R. Shelton. Sampling for approximate inference in continuous time Bayesian networks. In *Proceedings of Tenth International Symposium on Artificial Intelligence and Mathematics*, 2008.

Robert M. Fung and Kuo-Chu Chang. Weighing and integrating evidence for stochastic simulation in Bayesian networks. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pages 209–220, 1989.

Simon Godsill, Arnaud Doucet, and Mike West. Monte Carlo smoothing for non-linear time series. *Journal of the American Statistical Association*, 99:156–168, 2004.

Ralf Herbrich, Thore Graepel, and Brendan Murphy. Structure from failure. In *Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques*, pages 1–6, 2007.

Tim Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194, 1995.

Kin Fai Kan and Christian R. Shelton. Solving structured continuous-time Markov decision processes. In *Proceedings of Tenth International Symposium on Artificial Intelligence and Mathematics*, 2008.

Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 362–369, 2001.

Brenda Ng, Avi Pfeffer, and Richard Dearden. Continuous time particle filtering. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1360–1365, 2005.

Uri Nodelman and Eric Horvitz. Continuous time Bayesian networks for inferring users' presence and activities with extensions for modeling and evaluation. Technical Report MSR-TR-2003-97, Microsoft Research, December 2003.

Uri Nodelman, Christian R. Shelton, and Daphne Koller. Continuous time Bayesian networks. In *Proceedings of the Eighteenth International Conference on Uncertainty in Artificial Intelligence*, pages 378–387, 2002.

Uri Nodelman, Christian R. Shelton, and Daphne Koller. Learning continuous time Bayesian networks. In *Proceedings of the Nineteenth International Conference on Uncertainty in Artificial Intelligence*, pages 451–458, 2003.

Uri Nodelman, Daphne Koller, and Christian R. Shelton. Expectation propagation for continuous time Bayesian networks. In *Proceedings of the Twenty-First International Conference on Uncertainty in Artificial Intelligence*, pages 431–440, 2005a.

Uri Nodelman, Christian R. Shelton, and Daphne Koller. Expectation maximization and complex duration distributions for continuous time Bayesian networks. In *Proceedings of the Twenty-First International Conference on Uncertainty in Artificial Intelligence*, pages 421–430, 2005b.

James R. Norris. *Markov Chains*. Cambridge University Press, 1997.

Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, 1962.

Suchi Saria, Uri Nodelman, and Daphne Koller. Reasoning at the right time granularity. In *Proceedings of the Twenty-third Conference on Uncertainty in AI*, pages 421–430, 2007.

Ross D. Shachter and Mark A. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the Fifth International Conference on Uncertainty in Artificial Intelligence*, pages 221–234, 1989.

Christian R. Shelton, Yu Fan, William Lam, Joon Lee, and Jing Xu. Continuous time Bayesian network reasoning and learning engine. *Journal of Machine Learning Research*, 11(Mar):1137–1140, 2010.

Charles A. Sutton and Michael I. Jordan. Probabilistic inference in queueing networks. In Armando Fox and Sumit Basu, editors, *Proceedings of Third Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, 2008.

Greg C. G. Wei and Martin A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990.

Jing Xu and Christian R. Shelton. Continuous time Bayesian networks for host level network intrusion detection. In *European Conference on Machine Learning*, pages 613–627, 2008.

Journal of Machine Learning Research 11 (2010) 2141-2167 Submitted 11/09; Revised 4/10; Published 8/10

Matched Gene Selection and Committee Classifier for Molecular Classification of Heterogeneous Diseases

Guoqiang Yu YUG@VT.EDU
Yuanjian Feng YJFENG@VT.EDU
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Arlington, VA 22203, USA

David J. Miller DJMILLER@ENGR.PSU.EDU
Department of Electrical Engineering
The Pennsylvania State University
University Park, PA 16802, U.S.A

Jianhua Xuan XUAN@VT.EDU
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Arlington, VA 22203, USA

Eric P. Hoffman EHOFFMAN@CNMCRESEARCH.ORG
Research Center for Genetic Medicine
Children's National Medical Center
Washington, DC 20010, USA

Robert Clarke CLARKEL@GEORGETOWN.EDU
Department of Oncology, Physiology and Biophysics
Georgetown University
Washington, DC 20057, USA

Ben Davidson BEN.DAVIDSON@RADIUMHOSPITALET.NO
Department of Pathology
Radiumhospitalet-Rikshospitalet Medical Center
Montebello N-0310 Oslo, Norway

Ie-Ming Shih ISHIH@JHMI.EDU
Departments of Pathology and Gynecology and Oncology
Johns Hopkins Medical Institutions
Baltimore, MD 21231, USA

Yue Wang YUEWANG@VT.EDU
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Arlington, VA 22203, USA

Editor: Donald Geman

Abstract

Microarray gene expressions provide new opportunities for molecular classification of heterogeneous diseases. Although various reported classification schemes show impressive performance, most existing gene selection methods are suboptimal and are not well-matched to the unique charac-

teristics of the multicategory classification problem. Matched design of the gene selection method and a committee classifier is needed for identifying a small set of gene markers that achieve accurate multicategory classification while being both statistically reproducible and biologically plausible. We report a simpler and yet more accurate strategy than previous works for multicategory classification of heterogeneous diseases. Our method selects the union of one-versus-everyone (OVE) *phenotypic up-regulated* genes (PUGs) and matches this gene selection with a one-versus-rest support vector machine (OVRSVM). Our approach provides even-handed gene resources for discriminating both neighboring and well-separated classes. Consistent with the OVRSVM structure, we evaluated the fold changes of OVE gene expressions and found that only a small number of high-ranked genes were required to achieve superior accuracy for multicategory classification. We tested the proposed PUG-OVRSVM method on six real microarray gene expression data sets (five public benchmarks and one in-house data set) and two simulation data sets, observing significantly improved performance with lower error rates, fewer marker genes, and higher performance sustainability, as compared to several widely-adopted gene selection and classification methods. The MATLAB toolbox, experiment data and supplement files are available at http://www.cbil.ece.vt.edu/software.htm.

Keywords: microarray gene expression, multiclass gene selection, phenotypic up-regulated gene, multicategory classification

1. Background

The rapid development of gene expression microarrays provides an opportunity to take a genome-wide approach for disease diagnosis, prognosis, and prediction of therapeutic responsiveness (Clarke et al., 2008; Wang et al., 2008). When the molecular signature is analyzed with pattern recognition algorithms, new classes of disease are identified and new insights into disease mechanisms and diagnostic or therapeutic targets emerge (Clarke et al., 2008). For example, many studies demonstrate that global gene expression profiling of human tumors can provide molecular classifications that reveal distinct tumor subtypes not evident by traditional histopathological methods (Golub et al., 1999; Ramaswamy et al., 2001; Shedden et al., 2003; Wang et al., 2006).

While molecular classification falls neatly within supervised pattern recognition, high gene dimensionality and paucity of microarray samples pose challenges for, and inspire novel developments in classifier design and gene selection methodologies (Wang et al., 2008). For multicategory classification using gene expression data, various classifiers have been proposed and have achieved promising performance, including k-Nearest Neighbor Rule (kNN) (Golub et al., 1999), artificial neural networks (Wang et al., 2006), Support Vector Machine (SVM) (Ramaswamy et al., 2001), Naïve Bayes Classifier (NBC) (Liu et al., 2002), Weighted Votes (Tibshirani et al., 2002), and Linear Regression (Fort and Lambert-Lacroix, 2005). Many comparative studies show that SVM based classifiers outperform other methods on most bench-mark microarray data sets (Li et al., 2004; Statnikov et al., 2005).

An integral part of classifier design is gene selection, which can improve both classification accuracy and diagnostic economy (Liu et al., 2002; Shi et al., 2008; Wang et al., 2008). Many microarray-based studies suggest that, irrespective of the classification method, gene selection is vital for achieving good generalization performance (Statnikov et al., 2005). For multicategory classification using gene expression data, the criterion function for gene selection should possess high sensitivity and specificity, well match the specific classifiers used, and identify gene markers that are both statistically reproducible and biologically plausible (Shi et al., 2008; Wang et al., 2008). There are limitations associated with existing gene selection methods (Li et al., 2004; Statnikov et al., 2005). While wrapper methods consider joint discrimination power of a gene subset, complex clas-

sifiers used in wrapper algorithms for small sample size may overfit, producing non-reproducible gene subsets (Li et al., 2004; Shi et al., 2008). Moreover, discernment of the (biologically plausible) gene interactions retained by wrapper methods is often difficult due to the black-box nature of most classifiers (Shedden et al., 2003).

Conversely, most filtering methods for multicategory classification are straightforward extensions of binary discriminant analysis. These methods are devised without well matching to the classifier that is used, which typically leads to suboptimal classification performance (Statnikov et al., 2005). Popular multicategory filtering methods (which are extensions of two-class methods) include Signal-to-Noise Ratio (SNR) (Dudoit et al., 2002; Golub et al., 1999), Student's t-statistics (Dudoit et al., 2002; Liu et al., 2002), the ratio of Between-groups to Within-groups sum of squares (BW) (Dudoit et al., 2002), and SVM based Recursive Feature Elimination (RFE) (Li and Yang, 2005; Ramaswamy et al., 2001; Zhou and Tuck, 2007). However, as pointed out by Loog et al. (2001) in proposing their weighted Fisher criterion (wFC), simple extensions of binary discriminant analysis to multicategory gene selection are suboptimal because they overemphasize large between-class distances, that is, these methods choose gene subsets that preserve the distances of (already) well-separated classes, without reducing (and possibly with increase in) the large overlap between neighboring classes. This observation and the application of wFC to multicategory classification are further evaluated experimentally by Wang et al. (2006) and Xuan et al. (2007).

The work most closely related to our gene selection scheme is that of Shedden et al. (2003). These investigators focused on marker genes that are highly expressed in one phenotype relative to one or more different phenotypes and proposed a tree-based *one-versus-rest* (OVR) fold change evaluation between mean expression levels. The potential limitation here is that the criterion function considers the "rest of the classes" as a "super class", and thus may select genes that can distinguish a single class from the remaining super class, yet without giving any benefit in discriminating between classes within the super class. Such genes may compromise multicategory classification accuracy, especially when a small gene subset is chosen.

It is also important to note that, while univariate or multivariate analysis methods using complex criterion functions may reveal subtle marker effects (Cai et al., 2007; Liu et al., 2005; Xuan et al., 2007; Zhou and Tuck, 2007), they are also prone to overfitting. Recent studies have found that for small sample sizes, univariate methods fared comparably to multivariate methods (Lai et al., 2006; Shedden et al., 2003) and simple fold change analysis produced more reproducible marker genes than significance analysis of variance-incorporated t-tests (Shi et al., 2008).

In this paper, we propose matched design of the gene selection mechanism and a committee classifier for multicategory molecular classification using microarray gene expression data. A key feature of our approach is to match a simple *one-versus-everyone* (OVE) gene selection scheme to the OVRSVM committee classifier (Ramaswamy et al., 2001). We focus on marker genes that are highly expressed in one phenotype relative to each of the remaining phenotypes, namely Phenotypic Up-regulated Genes (PUGs). PUGs are identified using the fold change ratio computed between the specified phenotype mean and each of the remaining phenotype means. Thus, we consider a gene to be a marker for the specified phenotype if the average expression associated with this phenotype is high relative to the average expressions in each of the other phenotypes. To assure evenhanded resources for discriminating both neighboring and well-separated classes, we use a fixed number of PUGs for each phenotypic class and pool all phenotype-specific PUGs together to form a gene marker subset used by the OVRSVM committee classifier. All PUGs referenced by the committee classifier are individually interpretable as potential markers for phenotypic classes, allowing each

gene to inform the classifier in a way that is consistent with its mechanistic role (Shedden, et al., 2003). Since PUGs are the union of subPUGs selected by simple univariate OVE fold change analysis, they are expected to be statistically reproducible (Lai et al., 2006; Shedden et al., 2003; Shi et al., 2008).

We tested PUG-OVRSVM on five publicly available benchmarks and one in-house microarray gene expression data set and on two simulation data sets, observing significantly improved performance with lower error rates, fewer marker genes, and higher performance stability, as compared to several widely-adopted gene selection and classification methods. The reference gene selection methods are OVRSNR (Golub et al., 1999), OVRt-stat (Liu et al., 2002), pooled BW (Dudoit et al., 2002), and OVRSVM-RFE (Guyon et al., 2002), and the reference classifiers are kNN, NBC, and one-versus-one (OVO) SVM. With accuracy estimated by leave-one-out cross-validation (LOOCV) (Hastie et al., 2001), our experimental results show that PUG-OVRSVM outperforms all combinations of the above referenced gene selection and classification methods in the two simulation data sets and 5 out of the 6 real microarray gene expression data sets, and produces comparable performance on the one remaining data set. Specifically, tested on the widely-used benchmark microarray gene expression data set "multicategory human cancers data" (GCM) (Ramaswamy et al., 2001; Statnikov et al., 2005), PUG-OVRSVM produces a lower error rate of 11.05% (88.95% correct classification rate) than the best known benchmark error rate of 16.72% (83.28% correct classification rate) (Cai et al., 2007; Zhou and Tuck, 2007).

2. Methods

In this section, we first discuss multicategory classification and associated feature selection, with an emphasis on OVRSVM and application to gene selection for the microarray domain. This discussion then naturally leads to our proposed PUG-OVRSVM scheme.

2.1 Maximum a Posteriori Decision Rule

Classification of heterogeneous diseases using gene expression data can be considered a Bayesian hypothesis testing problem (Hastie et al., 2001). Let $\mathbf{x}_i = [x_{i1}, ..., x_{ij}, ..., x_{id}]$ be the real-valued gene expression profile associated with sample i across d genes for $i = 1, \ldots, N$ and $j = 1, \ldots, d$. Assume that the sample points $\mathbf{x}_i$ come from M classes, and denote the class conditional probability density function and class prior probability by $p(\mathbf{x}_i \mid \omega_k)$ and $P(\omega_k)$, respectively, for $k = 1, \ldots, M$. To minimize the Bayes risk averaged over all classes, the optimum classifier uses the well-known maximum a posteriori (MAP) decision rule (Hastie et al., 2001). Based on Bayes' rule, the class posterior probability for a given sample $\mathbf{x}_i$ is

$$P(\omega_k \mid \mathbf{x}_i) = \frac{P(\omega_k)\, p(\mathbf{x}_i \mid \omega_k)}{\sum_{k'=1}^{M} P(\omega_{k'})\, p(\mathbf{x}_i \mid \omega_{k'})}$$

and is used to (MAP) classify $\mathbf{x}_i$ to ω_k when

$$P(\omega_k \mid \mathbf{x}_i) > P(\omega_l \mid \mathbf{x}_i) \tag{1}$$

for all $l \neq k$.

2.2 Supervised Learning and Committee Classifiers

Practically, multicategory classification using the MAP decision rule can be approximated using parameterized discriminant functions that are trained by supervised learning. Let $f_k(\mathbf{x}_i, \theta)$, $k = 1, 2, \ldots, M$, be the M outputs of a machine classifier designed to discriminate between M classes (>2), where θ represents the set of parameters that fully specify the classifier, and with the output values assumed to be in the range $[0, 1]$. The desired output of the classifier will be "1" for the class to which the sample belongs and "0" for all other classes. Suppose that the classifier parameters are selected based on a training set so as to minimize the mean squared error (MSE) between the outputs of the classifier and the desired (class target) outputs,

$$\text{MSE} = \frac{1}{N} \sum_{k=1}^{M} \sum_{\mathbf{x}_i \in \omega_k} \left[[f_k(\mathbf{x}_i, \theta) - 1]^2 + \sum_{l \neq k} f_l^2(\mathbf{x}_i, \theta) \right]. \tag{2}$$

Then, it can be shown that the classifier is being trained to approximate the posterior probability for class ω_k given the observed $\mathbf{x}_i$, that is, the classifier outputs will converge to the true posterior class probabilities

$$f_k(\mathbf{x}_i, \theta) \rightarrow P(\omega_k \mid \mathbf{x}_i)$$

if we allow the classifier to be arbitrarily complex and if N is made sufficiently large. This result is valid for any classifier trained with the MSE criterion, where the parameters of the classifier are adjusted to simultaneously approximate M discriminant functions $f_k(\mathbf{x}_i, \theta)$ (Gish, 1990).

While there are numerous machine classifiers that can be used to implement the MAP decision rule (1) (Hastie et al., 2001), a simple yet elegant way of discriminating between M classes, and which we adopt here, is based on an OVRSVM committee classifier (Ramaswamy et al., 2001; Rifkin and Klautau, 2002; Statnikov et al., 2005). Intuitively, each term within the sum over k in (2) corresponds to an OVR binary classification problem and can be effectively minimized by suitable training of a binary classifier (discriminating class k from all other classes). By separately minimizing the MSE associated with each term in (2) via binary classifier training and, thus, effectively minimizing the total MSE, a set of discriminant functions $\{f_k(\mathbf{x}_i, \theta_k \subseteq \theta)\}$ can be constructed which, given a new sample point, apply the decision rule (1), but with $f_k(\mathbf{x}_i, \theta)$ playing the role of the posterior probability.

Among the great variety of binary classifiers that use regularization to control the capacity of the function spaces they operate in, the best known example is the SVM (Hastie et al., 2001; Vapnik, 1998). To carry over the advantages of regularization approaches for binary classification tasks to multicategory classification, the OVRSVM committee classifier uses M different SVM binary classifiers, each one separately trained to distinguish the samples in a single class from the samples in all remaining classes. For classifying a new sample point, the M SVMs are run, and the SVM that produces the largest (most positive) output value is chosen as the "winner" (Ramaswamy et al., 2001). For more detailed discussion, see the critical review and experimental comparison by Rifkin and Klautau (2002). Figure 1 shows an illustrative OVRSVM committee classifier for three classes. The OVRSVM committee classifier has proved highly successful at multicategory classification tasks involving finite or limited amounts of high dimensional data in real-world applications. OVRSVM produces results that are often at least as accurate as other more complicated methods including single machine multicategory schemes (Statnikov et al., 2005). Perhaps more importantly for our purposes, the OVR scheme can be matched with an OVE gene selection method, as we elaborate next.

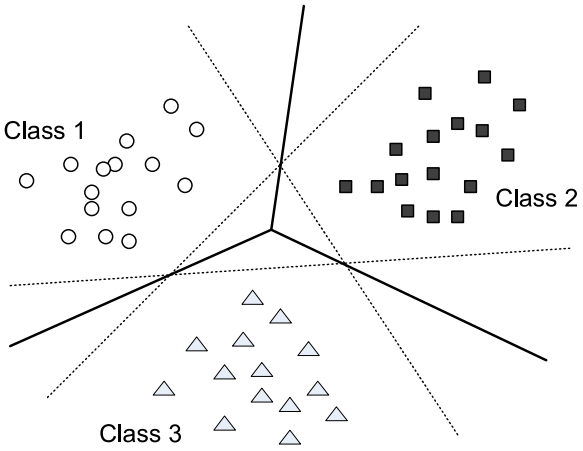


Figure 1: Conceptual illustration of OVR committee classifier for multicategory classification (three classes, in this case). The dotted lines are the decision hyperplanes associated with each of the component binary SVMs and the bold line-set represents the final decision boundary after the winner-take-all classification rule is applied.

2.3 One-Versus-Everyone Fold-change Gene Selection

While gene selection is vital for achieving good generalization performance (Guyon et al., 2002; Statnikov et al., 2005), perhaps even more importantly, the identified genes, if statistically reproducible and biologically plausible, are "markers", carrying information about the disease phenotype (Wang et al., 2008). We will propose two novel, effective gene selection methods for multicategory classification that are well-matched to OVRSVM committee classifiers, namely, OVR and OVE fold-change analyses.

OVR fold-change based PUG selection follows directly from the OVRSVM scheme. Let N_k be the number of sample points belonging to phenotype k; the geometric mean of the expression levels (on the untransformed scale) for gene j under phenotype k is

$$\mu_j(k) = \sqrt[N_k]{\prod_{i\in\omega_k} x_{ij}}$$

$j = 1,\ldots,d$; $k = 1,\ldots,M$. Then, we define the OVRPUGs as:

$$\mathbb{J}_{\text{PUG}} = \bigcup_{k=1}^{M} \mathbb{J}_{\text{PUG}}(k) = \bigcup_{k=1}^{M} \left\{ j \left| \frac{\mu_j(k)}{\sqrt[M-1]{\prod_{l\neq k}\mu_j(l)}} \geqslant \tau_k \right. \right\} \tag{3}$$

where $\{\tau_k\}$ are pre-defined thresholds chosen so as to select a fixed (equal) number of PUGs for each phenotype k. This PUG selection scheme (3) is similar to what has been previously proposed by Shedden et al. (2003):

$$\mathbb{J}_{\text{PUG}} = \bigcup_{k=1}^{M} \mathbb{J}_{\text{PUG}}(k) = \bigcup_{k=1}^{M} \left\{ j \left| \frac{\mu_j(k)}{\sqrt[N-N_k]{\prod_{i\notin\omega_k} x_{ij}}} \geqslant \tau_k \right. \right\}. \tag{4}$$

The critical difference between (3) and (4) is that the denominator term in (3) is the overall geometric center of the "geometric centers" associated with each of the remaining phenotypes while

the denominator term in (4) is the geometric center of all sample points belonging to the remaining phenotypes. When $\{N_k\}$ are significantly imbalanced for different k, the denominator term in (4) will be biased toward the dominant phenotype(s).

However, a problem associated with both PUG selection schemes specified by (3) and (4) (and with the OVRSNR criterion Golub et al., 1999) is that the criterion function considers the remaining classes as a single super class, which is suboptimal because it ignores a gene's ability to discriminate between classes *within* the super class.

We therefore propose OVE fold-change based PUG selection to fully support the objective of multicategory classification. Specifically, the OVEPUGs are defined as:

$$\mathbb{J}_{\mathrm{PUG}} = \bigcup_{k=1}^{M} \mathbb{J}_{\mathrm{PUG}}(k) = \bigcup_{k=1}^{M} \left\{ j \left| \frac{\mu_j(k)}{\max_{l \neq k} \{\mu_j(l)\}} \geqslant \tau_k \right. \right\} \tag{5}$$

where the denominator term is the maximum phenotypic mean expression level over the remaining phenotype classes. This seemingly technical modification turns out to have important consequences since it assures that the selected PUGs are highly expressed in one phenotype relative to *each* of the remaining phenotypes, that is, "high" (up-regulated) in phenotype k and "low" (down-regulated) in *all* phenotypes $l \neq k$. In our experimental results, we will demonstrate that (5) leads to better classification accuracy than (4) on a well-known multi-class cancer domain.

Adopting the same strategy as in Shedden et al. (2003), to assure even-handed gene resources for discriminating both neighboring and well-separated classes, we select a fixed (common) number of top-ranked phenotype-specific subPUGs for each phenotype, that is, $\|\mathbb{J}_{\mathrm{PUG}}(k)\| = N_{\mathrm{subPUG}}$ for all k, and pool all these subPUGs together to form the final gene marker subset $\mathbb{J}_{\mathrm{PUG}}$ for the OVRSVM committee classifier. In our experiments, the optimum number of PUGs per phenotype, N_{subPUG}, is determined by surveying the curve of classification accuracy versus N_{subPUG} and selecting the number that achieves the best classification performance. More generally, in practice, N_{subPUG} can be chosen via a cross validation procedure. Figure 2 shows the geometric distribution of the selected PUGs specified by (5), where the PUGs (highlighted data points) constitute the lateral-edge points of the convex pyramid defined by the scatter plot of the phenotypic mean expressions (Zhang et al., 2008). Different from the PUG selection schemes given by (3) and (4), the PUGs selected based on (5) are most compact yet informative, since the down-regulated genes that are not differentially expressed between the remaining phenotypes (the genes on the lateral faces of the scatter plot convex pyramid) are excluded. From a statistical point of view, extensive studies on the normalized scatter plot of microarray gene expression data by many groups including our own indicate that the PUGs selected by (5) approximately follow an independent multivariate super-Gaussian distribution (Zhao et al., 2005) where subPUGs are mutually exclusive and phenotypic gene expression patterns defined over the PUGs are statistically independent (Wang et al., 2003).

It is worth noting that the PUG selection by (5) also adopts a univariate fold-change evaluation that does not require calculation of either expression variance or of correlation between genes (Shi et al., 2008). For the small sample size case typical of microarray data, multivariate gene selection schemes may introduce additional uncertainty in estimating the correlation structure (Lai et al., 2006; Shedden et al., 2003) and thus may fail to identify true gene markers (Wang et al., 2008). The exclusion of the variance in our criterion is also supported by the variance stabilization theory (Durbin et al., 2002; Huber et al., 2002), because the geometric mean in (5) is equivalent to the arithmetic mean after logarithmic transformation and the gene expression after logarithmic transfor-

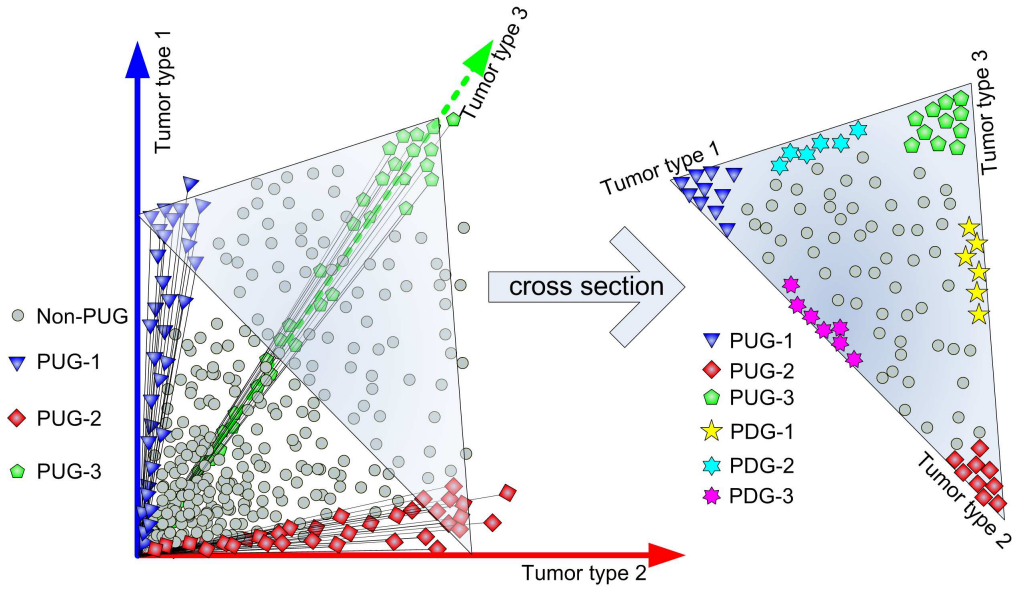


Figure 2: Geometric illustration of the selected one-versus-everyone phenotypic upregulated genes (OVEPUGs) associated with three phenotypic classes. Three-dimensional geometric distribution (on the untransformed scale) of the selected OVEPUGs, which reside around the lateral-edges of the phenotypic gene expression scatter plot convex pyramid, is shown in the left subfigure. A projected distribution of the selected OVEPUGs together with OVEPDGs is shown in the right cross-sectional plot, where OVEPDGs reside along the face-edges of the cross-sectional triangle.

mation approximately has the equal variance across different genes, especially for the up-regulated genes.

Corresponding to the definition of OVEPUGs, the OVEPDGs (which are down-regulated in one class while being up-regulated in all other classes) can be defined by the following criterion:

$$\mathbb{J}_{\mathrm{PDG}} = \bigcup_{k=1}^{M} \mathbb{J}_{\mathrm{PDG}}(k) = \bigcup_{k=1}^{M} \left\{ j \left| \frac{\min_{l \neq k} \{\mu_j(l)\}}{\mu_j(k)} \geqslant \tau_k \right. \right\}. \tag{6}$$

Furthermore, the combination of PUGs and PDGs can be defined as:

$$\mathbb{J}_{\mathrm{PUG+PDG}} = \bigcup_{k=1}^{M} \mathbb{J}_{\mathrm{PUG+PDG}}(k) = \bigcup_{k=1}^{M} \left\{ j \left| \max \left\{ \frac{\mu_j(k)}{\max_{l \neq k} \{\mu_j(l)\}}, \frac{\min_{l \neq k} \{\mu_j(l)\}}{\mu_j(k)} \right\} \geqslant \tau_k \right. \right\}. \tag{7}$$

Purely from the machine learning view, PDGs have the theoretical capability of being as discriminating as PUGs. Thus, PDGs merit consideration as candidate genes. However, there are several critical differences, with consequential implications, between lowly-expressed genes and highly-expressed genes, such as the extraordinarily large proportion and relatively large noise of the lowly-expressed genes. We have evaluated the classification performance of PUGs, PDGs, and

PUGs+PDGs, respectively. Experimental results show that PDGs have less discriminatory power than PUGs and the inclusion of PDGs actually worsens classification accuracy, compared to just using PUGs. Experiments and further discussion will be given in the results section.

2.4 Review of Relevant Gene Selection Methods

Here we briefly review four benchmark gene selection methods that have been previously proposed for multicategory classification, namely, OVRSNR (Golub et al., 1999), OVR t-statistic (OVRt-stat) (Liu et al., 2002), BW (Dudoit et al., 2002), and SVMRFE (Guyon et al., 2002).

Let $\mu_{j,k}$ and $\mu_{j,-k}$ be the arithmetic means of the expression levels of gene j associated with phenotype k and associated with the super class of remaining phenotypes, respectively, on the log-transformed scale, with $\sigma_{j,k}$ and $\sigma_{j,-k}$ the corresponding standard deviations. OVRSNR gene selection for multicategory classification is given by:

$$\mathbb{J}_{\text{OVRSNR}} = \bigcup_{k=1}^{M} \mathbb{J}_{\text{OVRSNR}}(k) = \bigcup_{k=1}^{M} \left\{ j \left| \frac{\left|\mu_{j,k} - \mu_{j,-k}\right|}{\sigma_{j,k} + \sigma_{j,-k}} \geqslant \tau_k \right. \right\}, \tag{8}$$

where τ_k is a pre-defined threshold (Golub et al., 1999). To assess the statistical significance of the difference between $\mu_{j,k}$ and $\mu_{j,-k}$, OVRt-stat applies a test of the null hypothesis that the means of two assumed normally distributed measurements are equal. Accordingly, OVRt-stat gene selection is given by Liu et al. (2002):

$$\mathbb{J}_{\text{OVR}t\text{-stat}} = \bigcup_{k=1}^{M} \mathbb{J}_{\text{OVR}t\text{-stat}}(k) = \bigcup_{k=1}^{M} \left\{ j \left| \frac{\left|\mu_{j,k} - \mu_{j,-k}\right|}{\sqrt{\sigma_{j,k}^2 \Big/ N_k + \sigma_{j,-k}^2 \Big/ (N - N_k)}} \geqslant \tau_k \right. \right\}, \tag{9}$$

where the p-values associated with each gene may be estimated. As aforementioned, one limitation of the gene selection schemes (8) and (9) is that the criterion function considers the remaining classes as a single group. Another is that they both require variance estimation.

Dudoit et al. (2002) proposed a pooled OVO gene selection method based on the BW sum of squares across all paired classes. Specifically, BW gene selection is specified by

$$\mathbb{J}_{\text{BW}} = \left\{ j \left| \frac{\sum_{i=1}^{N} \sum_{k=1}^{M} \mathbf{1}_{\omega_k}(i) \left(\mu_{j,k} - \mu_j\right)^2}{\sum_{i=1}^{N} \sum_{k=1}^{M} \mathbf{1}_{\omega_k}(i) \left(x_{ij} - \mu_{j,k}\right)^2} \geqslant \tau \right. \right\}, \tag{10}$$

where μ_j is the global arithmetic center of gene j over all sample points and $\mathbf{1}_{\omega_k}(i)$ is the indicator function reflecting membership of sample i in class k. As pointed out by Loog et al. (2001), BW gene selection may only preserve the distances of already well-separated classes rather than neighboring classes.

From a dimensionality reduction point of view, Guyon et al. (2002) proposed a feature subset ranking criterion for linear SVMs, dubbed the SVMRFE. Here, one first trains a linear SVM classifier on the full feature space. Features are then ranked based on the magnitude of their weights and are eliminated in the order of increasing weight magnitude. A widely adopted reduction strategy is to eliminate a fixed or decreasing percentage of features corresponding to the bottom portion of the ranked weights and then to retrain the SVM on the reduced feature space. Application to microarray gene expression data shows that the genes selected matter more than the classifiers with which they are paired (Guyon et al., 2002).

3. Results

We tested PUG-OVRSVM on five benchmarks and one in-house real microarray data set, and compared the performance to several widely-adopted gene selection and classification methods.

3.1 Description of the Real Data Sets

The numbers of samples, phenotypes, and genes, as well as the microarray platforms used to generate these gene expression data sets, are briefly summarized in Supplementary Tables 1∼7. The six data sets are the MIT 14 Global Cancer Map data set (GCM) (Ramaswamy et al., 2001), the NCI 60 cancer cell lines data set (NCI60) (Staunton et al., 2001), the University of Michigan cancer data set (UMich) (Shedden et al., 2003), the Central Nervous System tumors data set (CNS) (Pomeroy et al., 2002), the Muscular Dystrophy data set (MD) (Bakay et al., 2006), and the Norway Ascites data set (NAS). To assure a meaningful and well-grounded comparison, we emphasized data quality and suitability in choosing these test data sets. For example, the data sets cannot be too "simple" (if the classes are well-separated, all methods perform equally well) or too "complex" (no method will then perform reasonably well), and each class should contain sufficient samples to support some form of cross-validation assessment.

We also performed several important pre-processing steps widely adopted by other researchers (Guyon et al., 2002; Ramaswamy et al., 2001; Shedden et al., 2003; Statnikov et al., 2005). When the expression levels in the raw data take negative values, probably due to global probe-set calls and/or data normalization procedures, these negative values are replaced by a fixed small quantity (Shedden et al., 2003). On the log-transformed scale, we further conducted a variance-based unsupervised gene filtering operation to remove the genes whose expression standard deviations (across all samples) were less than a pre-determined small threshold; this effectively reduces the number of genes by half (Guyon et al., 2002; Shedden et al., 2003).

3.2 Experiment Design

We decoupled the two key steps of multicategory classification: 1) selecting an informative subset of marker genes and then 2) finding an accurate decision function. For the crucial first step we implemented five gene selection methods, including OVEPUG specified by (5), OVRSNR specified by (8), OVRt-stat specified by (9), pooled BW specified by (10), and SVMRFE described in Ramaswamy et al. (2001). We applied these methods to the six data sets, and for each data set, we selected a sequence of gene subsets with varying sizes, indexed by N_{subPUG}, the number of genes per class. In our experiments, this number was increased from 2 up to 100. There are several reasons why we do not go beyond 100 subPUGs per class. First, classification accuracy may be either flat or monotonically decreasing as the number of features increases beyond a certain point, due to the theoretical bias-variance dilemma. Second, even in some cases where best performance is achieved using all the gene features, the idea of feature selection is to find the minimum number of features needed to achieve good (near-optimal) classification accuracy. Third, when $N_{\text{subPUG}} = 100$, the total number of genes used for classification is already quite large (this number is maximized if the sets $\mathbb{J}_{\text{PUG}}(k)$ are mutually exclusive, in which case it is N_{subPUG} times the number of classes). Fourth, but not least important, a large feature reduction may be necessary not only complexity-wise, but also for interpreting the biological functions and pathway involvement when the selected PUGs are most relevant and statistically reproducible.

The quality of the marker gene subsets was then assessed by prediction performance on four subsequently trained classifiers, including OVRSVM, kNN, NBC, and OVOSVM. In relation to the proposed PUG-OVRSVM approach, we evaluated all combinations of these four different gene selection methods and three different classifiers on all six benchmark microarray gene expression data sets.

To properly estimate the accuracy of predictive classification, a validation procedure must be carefully designed, recognizing limits on the accuracy of estimated performance, in particular for small sample size. Clearly, classification accuracy must be assessed on labelled samples 'unseen' during training. However, for multicategory classification based on small, class-imbalanced data sets, single batch held-out test data may be precluded, as there will be insufficient samples for both accurate classifier training and accurate validation (Hastie et al., 2001). A practical alternative is a sound cross-validation procedure, wherein all the data are used for both training and testing, but with held-out samples in a testing fold not used for any phase of classifier training, including gene selection and classifier design (Wang et al., 2008). In our experiments, we chose LOOCV, wherein a test fold consists of a single sample; the rest of the samples are placed in the training set. Using only the training set, the informative genes are selected and the weights of the linear OVRSVM are fit to the data (Liu et al., 2005; Shedden et al., 2003; Yeang et al., 2001). It is worth noting that LOOCV is approximately unbiased, lessening the likelihood of misestimating the prediction error due to small sample size; however, LOOCV estimates do have considerable variance (Braga-Neto and Dougherty, 2004; Hastie et al., 2001). We evaluated both the lowest "sustainable" prediction error rate and the lowest prediction error rate, where the sequence of sustainable prediction error rates were determined based on a moving-average of error rates along the survey axis of the number of genes used for each class, N_{subPUG}, with a moving window of width 5. We also report the number of genes per class at which the best sustainable performance was obtained.

While the error rate is estimated through LOOCV and the optimum number of PUGs used per class is obtained by the aforementioned surveying strategy, we should point out that a two-level LOOCV could be applied to jointly determine the optimum N_{subPUG} and estimate the associated error rate; however, such an approach is computationally expensive (Statnikov et al., 2005). For the settings of structural parameters in the classifiers, we used $C = 1.0$ in the SVMs for all experiments (Vapnik, 1998), and chose $k = 1, 2, 3$ in kNNs under different training sample sizes per class, as recommended by Duda et al. (2001).

3.3 Experimental Results

Our first comparative study focused on the GCM data widely used for evaluating multicategory classification algorithms (Cai et al., 2007; Ramaswamy et al., 2001; Shedden et al., 2003; Zhou and Tuck, 2007). The performance curves of OVRSVM committee classifiers trained on the commonly pre-processed GCM data using the five different gene selection methods (OVEPUG, OVRSNR, OVRt-stat, BW, and SVMRFE) are detailed in Figure 3. It can be seen that our proposed OVEPUG selection significantly improved the overall multicategory classification when using different numbers of marker genes, as compared to the results produced by the four competing gene selection methods. For example, using as few as 9 genes per phenotypic class (with 126 distinct genes in total, that is, mutually exclusive PUGs for each class), we classified 164 of 190 (86.32%) of the tumors correctly. Furthermore, using LOOCV on the GCM data set of 190 primary malignant tumors, and using the optimal number of genes (61 genes per phenotypic class or 769 unique genes

in total), we achieved the best (88.95% or 169 of 190 tumors) sustainable correct predictions. In contrast, at its optimum performance, OVRSNR gene selection achieved 85.37% sustainable correct predictions using 25 genes per phenotypic class, OVRt-stat gene selection achieved 84.53% sustainable correct predictions using 71 genes per phenotypic class, BW gene selection achieved 80.53% sustainable correct predictions using 94 genes per phenotypic class, and SVMRFE gene selection achieved 84.74% sustainable correct predictions using 96 genes per phenotypic class. In our comparative study, instead of solely comparing the lowest error rates achieved by different gene selection methods, we also emphasized the sustainable correct prediction rates, as potential overfitting to the data may produce an (unsustainably) good prediction performance. For our experiments in Figure 3, based on the realistic assumption that the probability of good predictions purely "by chance" over a sequence of consecutive gene numbers is low, we defined the sustainable prediction/error rates based on the moving-averaged prediction/error rates over $\delta = 5$ consecutive gene numbers. Here, δ gives the sustainability requirement.

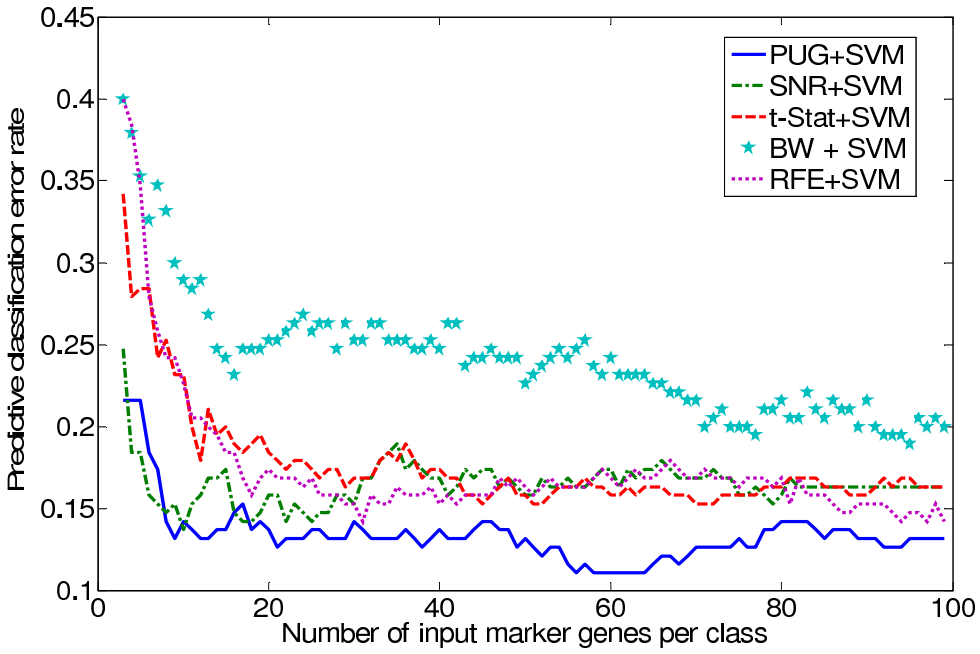


Figure 3: Comparative study on five gene selection methods (OVEPUG, OVRSNR, OVRt-stat, BW, and SVMRFE) using the GCM benchmark data set. The curves of classification error rates were generated by using OVRSVM committee classifiers with varying size of the input gene subset.

For the purpose of information sharing with readers, based on publicly reported optimal results for different methods, we have summarized in Table 1 the comparative performance achieved by PUG-OVRSVM and eight existing/competing methods on the benchmark GCM data set, along with the gene selection methods used, the chosen classifiers, sample sizes, and the chosen cross-validation schemes. Obviously, since the reported prediction error rates were generated by different algorithms and under different conditions, any conclusions based on simple/direct comparisons of the reported results must be carefully drawn. We have chosen not to independently reproduce results

by re-implementing the methods listed in Table 1, firstly because we typically do not have access to public domain code implementing other authors' methods and secondly because we feel that high reproducibility of previously published results may not be expected without knowing some likely undeclared optimization steps and/or additional control parameters used in the actual computer codes. Nevertheless, many reported prediction error rates on the GCM data set were actually based on the same/similar training sample set ($144 \sim 190$ primary tumors) and the LOOCV scheme used in our PUG-OVRSVM experiments; furthermore, it was reported that the prediction error rates estimated by LOOCV and 144/54 split/held-out test were very similar (Ramaswamy et al., 2001). Specifically, the initial work on GCM by Ramaswamy et al. (2001) reported an achieved 77.78% prediction rate, and some improved performance was later reported by Yeang et al. (2001) and Liu et al. (2002), achieving 81.75% and 79.99% prediction rates, respectively. In the work most closely related to our gene selection scheme by Shedden et al. (2003), using a kNN tree classifier and using OVR fold-change based gene selection specified by (4), a prediction rate of 82.63% was achieved. In relation to these reported results on GCM, as indicated in Table 1, our proposed PUG-OVRSVM method produced the best sustainable prediction rate of 88.95%.

References	Gene-select	Classifier	Sample	CV scheme	Error rate
Ramaswamy et al. (2001)	OVRSVM RFE	OVRSVM	144&198	LOOCV 144/54	22.22%
Yeang et al. (2001)	N/A	OVRSVM	144	LOOCV	18.75%
Ooi and Tan (2003)	Genetic algorithm	MLHD	198	144/54	18.00%
Shedden et al. (2003)	OVR fold-change	kNN Tree	190	LOOCV	17.37%
Liu et al. (2005)	Genetic algorithm	OVOSVM	N/A	LOOCV	20.01%
Statnikov et al. (2005)	No gene selection	CS-SVM	308	10-fold	23.40%
Zhou and Tuck (2007)	CS-SVM RFE	OVRSVM	198	4-fold	16.72%
Cai et al. (2007)	DISC-GS	kNN	190	144/46	21.74%
PUG-OVRSVM	PUG	OVRSVM	190	LOOCV	**11.05%**

Table 1: Summary of comparative performances by OVEPUG-OVRSVM and eight competing methods (based on publicly reported optimum results) on the GCM benchmark data set.

A more stringent evaluation of the robustness of a classification method is to carry out the predictions on multiple data sets and then assess the overall performance (Statnikov et al., 2005). Our second comparative study evaluated the aforementioned five gene selection methods using the six benchmark microarray gene expression data sets. To determine whether the genes selected matter more than the classifiers used (Guyon et al., 2002), we used a common OVRSVM committee classifier and LOOCV scheme in all the experiments, and summarized the corresponding results in Table 2. For each experiment that used a distinct gene selection scheme applied to a distinct data set, we reported both sustainable (with sustainability requirement $\delta = 5$) and lowest (within parentheses) prediction error rates, as well as the number of genes per class that were used to produce these results. Clearly, the selected PUGs based on (5) produced the highest overall sustainable prediction rates as compared to the other four competing gene selection methods. Specifically, PUG is the consistent winner in 22 of 24 competing experiments (combinations of four gene selection schemes and six testing data sets). It should be noted that although BW and OVRSNR achieved comparably low prediction error rates on the CNS data set (with relatively balanced mixture distributions), they

also produced high prediction error rates on the other testing data sets; the other competing gene selection methods also show some level of performance instability across data sets.

Gene-select	GCM	NCI60	UMich	CNS	MD	NAS
OVE PUG	**11.05%** (11.05%) [61 g/class]	**27.33%** (26.67%) [52 g/class]	**1.08%** (0.85%) [26 g/class]	**7.14%** (7.14%) [71 g/class]	**19.67%** (19.01%) [46 g/class]	**13.16%** (13.16%) [42 g/class]
OVR SNR	14.63% (13.68%) [25 g/class]	31.67% (31.67%) [58 g/class]	1.42% (1.42%) [62 g/class]	**7.14%** (7.14%) [57 g/class]	23.97% (23.97%) [85 g/class]	16.32% (15.79%) [54 g/class]
OVR t-stat	15.47% (15.26%) [71 g/class]	31.67% (31.67%) [56 g/class]	1.70% (1.70%) [45 g/class]	7.62% (7.14%) [92 g/class]	23.47% (22.31%) [56 g/class]	15.79% (15.79%) [74 g/class]
BW	19.47% (18.95%) [94 g/class]	31.67% (31.67%) [55 g/class]	1.30% (1.13%) [92 g/class]	**7.14%** (7.14%) [56 g/class]	19.83% (19.01%) [71 g/class]	21.05% (21.05%) [65 g/class]
SVM RFE	15.26% (14.21%) [96 g/class]	29.00% (28.33%) [81 g/class]	1.13% (1.13%) [58 g/class]	14.29% (14.29%) [53 g/class]	29.09% (28.10%) [73 g/class]	32.11% (31.58%) [94 g/class]

Table 2: Performance comparison between five different gene selection methods tested on six benchmark microarray gene expression data sets, where the predictive classification error rates for all methods were generated based on OVRSVM committee classification and an LOOCV scheme. Both sustainable and lowest (within parentheses) error rates are reported together with number of genes used per class.

To give more complete comparisons that also involved different classifiers (Statnikov et al., 2005), we further illustrate the superior prediction performance of the matched OVEPUG selection and OVRSVM classifier as compared to the best results produced by combinations of three different classifiers (OVOSVM, kNN, NBC) and four gene selection methods (PUG, OVRSNR, OVRt-stat, pooled BW). The optimum experimental results achieved over all combinations of these methods on the six data sets are summarized in Table 3, where we report both sustainable prediction error rates and the corresponding gene selection methods. Again, PUG-OVRSVM outperformed all other methods on all six data sets and was a clear winner in all 15 competing experiments. Our comparative studies also reveal that although gene selection is a critical step of multi-category classification, the classifiers used do indeed play an important role in achieving good prediction performance.

3.4 Comparison Results on the Realistic Simulation Data Sets

To more reliably validate and compare the performance of the different gene selection methods, we have conducted additional experiments involving realistic simulations. The advantage of using synthetic data is that, unlike the real data sets often with small sample size and with LOOCV as the only applicable validation method, large testing samples can be generated to allow an accurate and reliable assessment of a classifier's generalization performance. Two different simulation approaches were implemented. In both, we modeled the joint distribution for microarray data under each class and generated *i.i.d.* synthetic data sets consistent both with these distributions and with assumed class priors. In the first approach, we chose the class-conditional models consistent with commonly

	GCM	NCI60	UMich	CNS	MD	NAS
OVR SVM	**11.05%** (OVEPUG)	**27.33%** (OVEPUG)	**1.08%** (OVEPUG)	**7.14%** (OVEPUG)	**19.67%** (OVEPUG)	**13.16%** (OVEPUG)
OVO SVM	14.74% (OVEPUG)	33.33% (OVRSNR)	1.70% (OVEPUG)	9.52% (BW)	19.83% (BW)	16.32% (OVRSNR)
kNN	21.05% (OVEPUG)	31.67% (OVRt-stat)	2.27% (OVEPUG)	13.33% (OVEPUG)	21.81% (BW)	13.68% (OVRt-stat)
NBC	36.00% (OVRSNR)	51.67% (OVRSNR)	2.83% (OVRt-stat)	37.62% (BW)	37.69% (BW)	34.21% (OVEPUG)

Table 3: Performance comparison based on the lowest predictive classification error rates produced by OVEPUG-OVRSVM and the optimum combinations of five different gene selection methods and three different classifiers, tested on six benchmark microarray gene expression data sets and assessed via the LOOCV scheme.

accepted properties of microarray data (few discriminating features, many non-discriminating features, and with small sample size) (Hanczar and Dougherty, 2010; Wang et al., 2002). In the second approach, we directly estimated the class-conditional models based on a real microarray data set and then generated the *i.i.d.* samples according to the learned models.

3.4.1 DESIGN I

We simulated 5000 genes, with 90 "relevant" and 4910 "irrelevant" genes. Inspired by gene clustering concept in modelling local correlations, we divided the genes into 1000 blocks of size five, each containing exclusively either relevant or irrelevant genes. Within each block the correlation coefficient is 0.9, with zero correlation across blocks. Irrelevant genes are assumed to follow a (univariate) standard normal distribution, for all classes. Relevant genes also follow a normal distribution with variance 1 for all classes. There are three equally likely classes, A, B and C. The mean vectors of the 90 relevant genes under each class are shown in Table 4. The means were chosen to make the classification task neither too easy nor too difficult and to simulate unequal distances between the classes—A and B are relatively close, with C more distant from both A and B.

	The mean vector μ for each class
μ_A	[2.8 2.8 2.8 2.8 2.8 1 1 1 1 1 2 2 2 2 2 0.5 0.5 0.5 0.5 0.5 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0.5 0.5 0.5 0.5 0.5 0 0 0 0 0 1 1 1 1 1 3 3 3 3 3 0.1 0.1 0.1 0.1 0.1]
μ_B	[1 1 1 1 1 2.8 2.8 2.8 2.8 2.8 2 2 2 2 2 0.5 0.5 0.5 0.5 0.5 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0.5 0.5 0.5 0.5 0.5 0 0 0 0 0 1 1 1 1 1 3 3 3 3 3 0.1 0.1 0.1 0.1 0.1]
μ_C	[1 1 1 1 1 1 1 1 1 1 14.4 14.4 14.4 14.4 14.4 8.5 8.5 8.5 8.5 8.5 8 8 8 8 8 10 10 10 10 10 10 10 10 10 10 9 9 9 9 9 10 10 10 10 10 3 3 3 3 3 10 10 10 10 10 10 10 10 10 10 10 10 10 8.5 8.5 8.5 8.5 8.5 8 8 8 8 8 9 9 9 9 9 11 11 11 11 11 7.1 7.1 7.1 7.1 7.1]

Table 4: The mean vectors of the 90 relevant genes under each of the three classes.

We randomly generated 100 synthetic data sets, each partitioned into a small training set of 60 samples (20 per class) and a large testing set of 6000 samples.

3.4.2 DESIGN II

The second approach models each class as a more realistic multivariate normal distribution $N(\mu,\Sigma)$, with the class's mean vector μ and covariance matrix Σ directly learned from the real microarray data set GCM. Estimation of a covariance matrix is certainly a challenging task, specifically due to the very high dimensionality of the gene space ($p = 15,927$ genes in the GCM data) and only a few dozen samples available for estimating $p(p-1)/2$ free covariate parameters per class. It is also computationally prohibitive to generate random vectors based on full covariances on a general desktop computer. To address both of these problems, we applied a factor model (McLachlan and Krishnan, 2008), which can significantly reduces the number of free parameters to be estimated while capturing the main correlation structure in the data.

In factor analysis, the observed $p \times 1$ vector $\mathbf{t}$ is modeled as

$$\mathbf{t} = \mu + \mathbf{W}\mathbf{x} + \varepsilon,$$

where μ is the mean vector of observation $\mathbf{t}$, $\mathbf{W}$ is a $p \times q$ matrix of factor loadings, $\mathbf{x}$ is the $q \times 1$ latent variable vector with standard normal distribution $N(\mathbf{0},\mathbf{I})$ and ε is noise with independent multivariate normal distribution $N(\mathbf{0},\Psi)$, $\Psi = \mathbf{diag}\left(\sigma_1^2,\ldots,\sigma_p^2\right)$. The resulting covariance matrix Σ is

$$\Sigma = \mathbf{W}\mathbf{W}^T + \Psi.$$

Estimation of Σ reduces to estimating $\mathbf{W}$ and Ψ, totaling $p(q+1)$ parameters. Usually, we have q much less than p. The factor model is learned via the EM algorithm (McLachlan and Krishnan, 2008), initialized by probabilistic principal component analysis (Tipping and Bishop, 1999).

In our experiments, we set $q = 5$, which typically accounted for 60% of the energy. We also tried $q = 3$ and 7 and observed that the relative performance remained unchanged, although the absolute performance of all methods does change with q.

Five phenotypic classes were used in our simulation: breast cancer, lymphoma, bladder cancer, leukemia and CNS. 100 synthetic data sets were generated randomly according to the learned class models from the real data of these five cancer types. The dimension for each sample is 15,927. For each data set, the training sample size was the same as used in the real data experiments, with 11, 22, 11, 30, and 20 samples in the five respective classes; and the testing set consisted of 3,000 samples, 600 per class.

3.5 Evaluation of Performance

For a given gene-selection method and for each data set (indexed by $i = 1,\ldots,100$), the classifier F_i is learned. We then evaluate F_i on the i-th testing set, and measure the error rate ε_i. Since the testing set has quite large sample size, we would expect ε_i to be close to the true classification error rate for F_i. Over 100 simulation data sets, we then calculated both the average classification error $\bar{\varepsilon}$ and the standard deviation σ.

Furthermore, let $\varepsilon_{i,PUG}$ denote the error rate associated with PUGs on testing set i, and similarly, let $\varepsilon_{i,SNR}$, $\varepsilon_{i,t\text{-}stat}$, $\varepsilon_{i,BW}$ and $\varepsilon_{i,SVMRFE}$ denote the error rates associated with the four peer gene selection methods. The error rate difference between two methods, for example, PUG and SNR, is defined by

$$D_i(PUG,SNR) = \varepsilon_{i,PUG} - \varepsilon_{i,SNR}.$$

For each synthetic data set, we define the "winner" as the one with the least testing error rate. For each method, the mean and standard deviation of the error rate and the frequency of winning are examined for performance evaluation. In addition, the histogram of error rate differences between PUG and peer methods are provided.

3.6 Experimental Results on the Simulation Data Sets

We tested all gene selection methods using the common OVRSVM classifier. All the experiments were done using the same procedure as on the real data sets, except with LOOCV error estimation replaced by the error estimation using large size independent testing data. Figure 4, analogous to Figure 3 while on the realistic synthetic data whose model was estimated from GCM data set (simulation data under design II), shows the comparative study on five gene selection methods (OVEPUG, OVRSNR, OVRt-stat, BW, and SVMRFE). Tables 5 and 6 show the average error, standard deviation, and frequency of winning, estimated based on the 100 simulation data sets. PUG has the smallest average error over all competing methods. PUG also is the most stable method (with the smallest standard deviation). Tables 7 and 8 provide the comparison results of the five competing methods on the first ten data sets.

Figures 5 and 6 show histograms of the error difference between PUG and other methods, where a negative value of the difference indicates better performance by PUG. The red bar shows the position where the two methods are equal. We can see that the vast majority of differences are negative. Actually, as indicated in Tables 5 and 6, there is no positive difference in the subfigures of Figure 5 and at most one positive difference in the subfigures of Figure 6.

	PUG	SNR	t-stat	BW	SVMRFE
mean	**0.0724**	0.1129	0.1135	0.1165	0.1203
std deviation	**0.0052**	0.0180	0.0188	0.0177	0.0224
frequency of 'winner'	**100**	0	0	0	0

Table 5: The mean and standard deviation of classification error and the frequency of winner based on 100 simulation data sets with design I.

	PUG	SNR	t-stat	BW	SVMRFE
mean	**0.0712**	0.1311	0.1316	0.2649	0.0910
std deviation	**0.0201**	0.0447	0.0449	0.0302	0.0244
frequency of 'winner'	**99**	0	0	0	1

Table 6: The mean and standard deviation of classification error and the frequency of winner based on 100 simulation data sets with design II.

3.7 Comparison Between PUGs and PDGs

In this experiment, we selected PDGs according to the definition given in (6) and evaluated gene selection based on PUGs, PDGs, and based on their union, as given in (7). Again, all gene selection methods were coupled with the OVRSVM classifier. Table 9 shows classification performance for

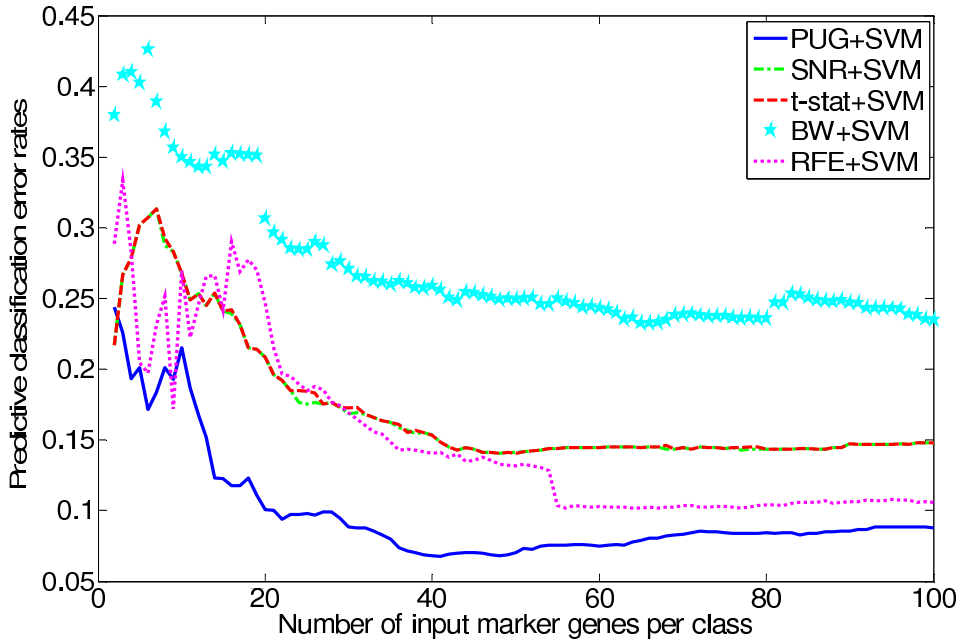


Figure 4: Comparative study on five gene selection methods (OVEPUG, OVRSNR, OVRt-stat, BW, and SVMRFE) on one simulation data set under design II. The curves of classification error rates were generated by using OVRSVM committee classifiers with varying size of the input gene subset.

	sim_1	sim_2	sim_3	sim_4	sim_5	sim_6	sim_7	sim_8	sim_9	sim_10
PUG	**0.0864**	**0.0773**	**0.0697**	**0.0681**	**0.0740**	**0.0761**	**0.0740**	**0.0721**	**0.0666**	**0.0758**
SNR	0.1078	0.1092	0.1028	0.1279	0.1331	0.1004	0.1011	0.1253	0.0817	0.0838
t-stat	0.1109	0.1089	0.1022	0.1251	0.1333	0.0991	0.1016	0.1268	0.0823	0.0832
BW	0.1127	0.0995	0.1049	0.1271	0.1309	0.1107	0.1044	0.1291	0.0903	0.0845
SVMRFE	0.1030	0.1009	0.0967	0.1219	0.1248	0.1016	0.1107	0.1191	0.1198	0.0933

Table 7: Comparison of the classification error for the first ten simulation data sets with design I.

PUGs, PDGs and PUGs+PDGs. Clearly, PDGs have less discriminatory power than PUGs, and the inclusion of PDGs (generally) worsens classification accuracy, compared with just using PUGs.

	sim_1	sim_2	sim_3	sim_4	sim_5	sim_6	sim_7	sim_8	sim_9	sim_10
PUG	**0.0694**	**0.0610**	**0.0748**	**0.0675**	0.0536	**0.0474**	**0.0726**	**0.0818**	**0.0560**	**0.0700**
SNR	0.1559	0.0659	0.1142	0.1211	0.0508	0.1937	0.1568	0.1464	0.0797	0.0711
t-stat	0.1559	0.0659	0.1142	0.1210	0.0508	0.1939	0.1568	0.1464	0.0797	0.0712
BW	0.2373	0.2698	0.2510	0.2650	0.3123	0.2464	0.3070	0.2236	0.2800	0.3055
SVMRFE	0.0906	0.0739	0.0864	0.0852	**0.0426**	0.0776	0.0863	0.0973	0.0655	0.0730

Table 8: Comparison of the classification error for the first ten simulation data sets with design II.

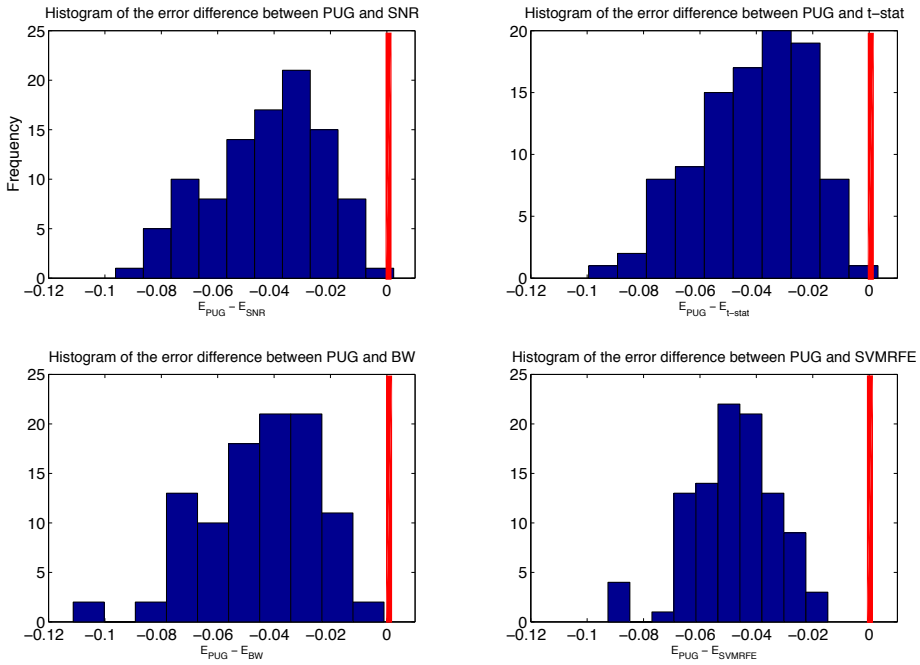


Figure 5: Histogram of the error difference between PUG and other methods with design I.

Error Rate	GCM	NCI60	UMich	CNS	MD	NAS
PUG	**11.05%**	**27.33%**	**1.08%**	**7.14%**	**19.67%**	**13.16%**
PDG	17.58%	30.33%	1.98%	9.52%	26.28%	25.79%
PUG+PDG	14.53%	30.67%	1.13%	**7.14%**	23.14%	15.79%

Table 9: Classification comparison of PUG and PDG on the six benchmark data sets.

There are several potential reasons that may jointly explain the non-contributing or even negative role of the included PDGs. First, the number of PDGs are much less than that of PUGs, that is, PUGs represent the significant majority of informative genes when PUGs and PDGs are jointly considered, as shown in Table 10 (Top PUG+PDGs were selected with 10 genes per class and we counted how many PUGs are included in the total). Second, PDGs are less reliable than PUGs due to the noise characteristics of gene expression data, that is, low gene expressions contain relatively large additive noise after log-transformation (Huber et al., 2002; Rocke and Durbin, 2001). This is further exacerbated by the follow-up one-versus-rest classifier because there are many more samples in the 'rest' group than in the 'one' group. This practically increases the relative noise/variability associated with PDGs in the 'one' group. In addition, PUGs are consistent with the practice of molecular pathology and thus may have broader clinical utility, for example, most currently available disease gene markers are highly expressed (Shedden et al., 2003).

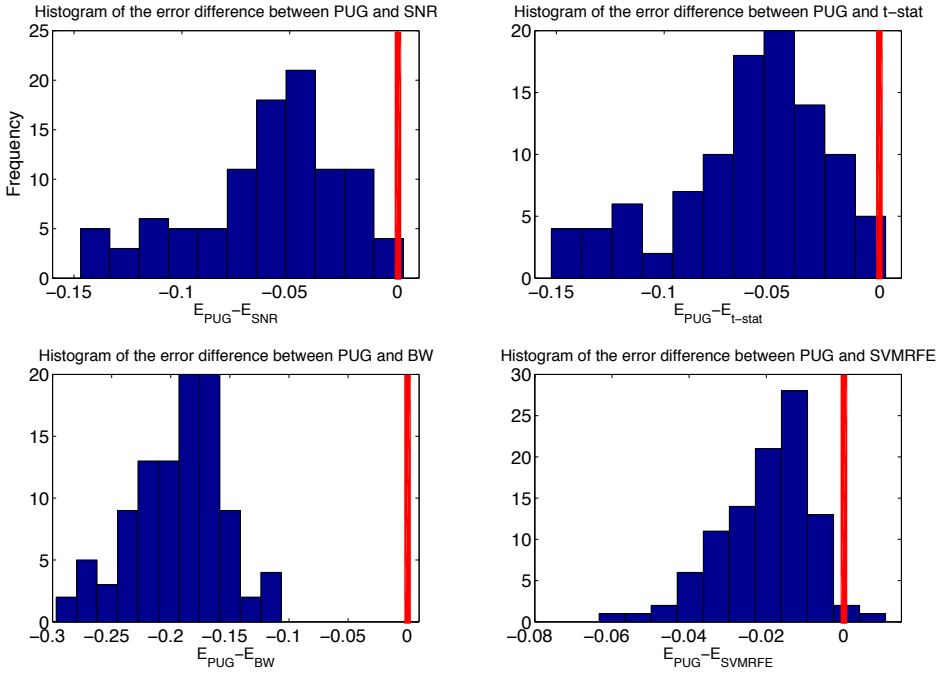


Figure 6: Histogram of the error difference between PUG and other methods with design II.

	GCM	NCI60	UMich	CNS	MD	NAS
No. of PUG	113	76	56	33	76	65
No. of PUG+PDG	140	90	60	50	130	70
% of PUG	80.71%	84.44%	93.33%	66.00%	58.46%	92.86%

Table 10: Classification comparison of PUG and PDG on the six benchmark data sets.

3.8 Marker Gene Validation by Biological Knowledge

We have applied existing biological knowledge to validate biological plausibility of the selected PUG markers for two data sets, GCM and NAS. The full list of genes most highly associated with each of the 14 tumor types in the GCM data set are detailed in the Supplementary Tables 8 and 9.

3.8.1 BIOLOGICAL INTERPRETATION FOR GCM DATA SET

Prolactin-induced protein, which is regulated by prolactin activation of its receptors, ranks highest among the PUGs associated with breast cancer. Postmenopausal breast cancer risk is strongly associated with elevated prolactin levels (PubMed IDs 15375001, 12373602, 10203283). Interestingly, prolactin release proportionally increases with increasing central fat in obese women (PubMed ID 15356045) and women with this pattern of obesity have an increased risk of breast cancer mortality (PubMed ID 14607804). Other genes of interest that rank among the top 10 breast cancer PUGs include CRABP2, which transports retinoic acid to the nucleus. Retinoids are important regulators of breast cell function and show activity as potential breast cancer chemopreventive agents (PubMed IDs 11250995, 12186376). Mammglobin is primarily expressed in normal breast epithelium and breast cancers (PubMed ID 12793902). Carbonic anhydrase XII is expressed in breast cancers and

is generally considered a marker of a good prognosis (PubMed ID 12671706). The selective expression and/or function of these genes in breast cancers are consistent with their selection as PUGs in the classification scheme.

The top 10 PUGs associated with prostate cancer include several genes strongly associated with the prostate including prostate specific antigen (PSA) and its alternatively spliced form 2, and prostatic secretory protein 57. The role of PSA gene KLK3 and KLK1 as a biomarker of prostate cancer is well established (PubMed ID 19213567). Increased NPY expression is associated with high-grade prostatic intraepithelial neoplasia and poor prognosis in prostate cancers (PubMed ID 10561252). ACPP is another prostate specific protein biomarker (PubMed ID 8244395). The strong representation of genes that show clear selectivity for expression within the prostate illustrates the potential of the PUGs as bio-markers linked to the biology of the underlying tissues.

Several of the selected PUG markers for uterine cancer fit very well with our current biological understanding of this disease. It is well-established that estrogen receptor alpha (ESR1) is expressed or amplified in human uterine cancer (PubMed IDs 18720455, 17911007, 15251938), while the Hox7 gene (MSX1) contributes to uterine function in cow and mouse models, especially at the onset of pregnancy (PubMed IDs 7908629, 14976223, 19007558). Mammaglobin 2 (SCGB2A1) is highly expressed in a specific type of well-differentiated uterine cancer (endometrial cancers) (PubMed ID 18021217), and PAM expression in the rat uterus is known to be regulated by estrogen (PubMed IDs 9618561, 9441675). Other PUGs provide novel insights into uterine cancer that are deserving of further study. Our PUG selection ranks HE4 higher than the well-established CA125 marker, which may suggest HE4 as a promising alternative for the clinical management of endometrial cancer. One recent study (PubMed ID 18495222) shows that, at 95% specificity, the sensitivity of differentiating between controls and all stages of uterine cancer is 44.9% using HE4 versus 25.2% using CA125 ($p = 0.0001$).

Osteopontin (OPN) is an integrin-binding protein that is involved in tumorigenesis and metastasis. OPN levels in the plasma of patients with ovarian cancer are much higher compared with plasma from healthy individuals (PubMed ID 11926891). OPN can increase the survival of ovarian cancer cells under stress conditions in vitro and can promote the late progression of ovarian cancer in vivo, and the survival-promoting functions of OPN are mediated through Akt activation (PubMed ID 19016748). Matrix metalloproteinase 2 (MMP2) is an enzyme degrading collagen type IV and other components of the basement membrane. MMP-2 is expressed by metastatic ovarian cancer cells and functionally regulates their attachment to peritoneal surfaces (PubMed ID 18340378). MMP2 facilitates the transmigration of human ovarian carcinoma cells across endothelial extracellular matrix (PubMed ID 15609323). Glutathione peroxidase 3 (GPX3) is one of several isoforms of peroxidases that reduce hydroperoxides to the corresponding alcohols by means of glutathione (GSH) (PubMed ID 17081103). GPX3 has been shown to be highly expressed in ovarian clear cell adenocarcinoma. Moreover, GPX3 has been associated with low cisplatin sensitivity (PubMed ID 19020706).

3.8.2 Biological Interpretation for NAS Data Set

Several top-ranking gene products identified by our computational method have been well established as tumor-type specific markers and many of them have been used in clinical diagnosis. For example, mucin 16, also known as CA125, is a FDA-approved serum marker to monitor disease progression and recurrence in ovarian cancer patients (PubMed ID 19042984). Likewise, kallikrein

family members including KLK6 and KLK8 are known to be ovarian cancer associated markers which can be detected in body fluids in ovarian cancer patients (PubMed ID 17303231). TITF1 (also known as TTF1) has been reported as a relatively specific marker in lung adenocarcinoma (PubMed ID 17982442) and it has been used to assist differential diagnosis of lung cancer from other types of carcinoma. Fatty acid synthase (FASN) is a well-known gene that is often upregulated in breast cancer (PubMed ID 17631500) and the enzyme is amenable for drug targeting using FASN inhibitors, suggesting that it can be used as a therapeutic target in breast cancer. The above findings indicate the robustness of our computational method in identifying tumor-type specific markers and in classifying different types of neoplastic diseases. Such information could be useful in translational studies (PubMed ID 12874019). Metastatic carcinoma of unknown origin is a relatively common presentation in cancer patients and an accurate diagnosis of the tumor type in the metastatic diseases is important to direct appropriate treatment and predict clinical outcome. The distinctive patterns of gene expression characteristic to various types of cancer may help pathologists and clinicians to better manage their patients.

3.9 Gene Comparisons Between Methods

It may be informative to provide some initial analysis on how the selected genes compare between methods; however, without definitive ground truth on cancer markers, the utility of this information is somewhat limited and should, thus, be treated as anecdotal, rather than conclusive. Specifically, we have now done some assessment of how differentially these gene selection methods rank some known cancer marker genes. The overlap rate is defined as the number of genes commonly selected by two methods over the maximum size of the two selected gene sets. Let G_1 and G_2 denote the gene sets selected by gene selection methods 1 and 2, respectively, and $|G|$ denote the cardinality (the size) of set G. The overlap rate between G_1 and G_2 is

$$R = \frac{|G_1 \cap G_2|}{\max(|G_1|, |G_2|)}.$$

Table 11 shows the overlap rate between methods on the top 100 genes per class. We can see that the overlap rates between methods are generally low except for the pair of SNR and t-stat. BW genes are quite different from the genes selected by all other methods and have only about 15% overlap rate with PUG and SVMRFE. The relatively high overlap rate between SNR and t-stat may be expected since they use quite similar summary statistics in their selection criteria.

We have also examined a total of 16 genes with known associations with 4 tumor types. These 16 genes are well-known markers supported by current biological knowledge. The rank of biomedical importance of these genes produced by each method is summarized in Table 12. When a gene is not listed in the top 100 genes by a wrapper method like SVMRFE, we simply assign the rank as '$>$100'. Generally but not uniformly across cancer types, these validated marker genes are highly ranked in the PUGs list as compared to other methods, and thus will be surely selected by PUG criterion.

4. Discussion

In this paper, we address several critical yet subtle issues in multicategory molecular classification applied to real-world biological and/or clinical applications. We propose a novel gene selection methodology matched to the multicategory classification approach (potentially with an unbalanced

Overlapping Rate	PUG	SNR	t-stat	BW	SVMRFE
PUG	1	0.4117	0.3053	0.1450	0.4057
SNR	0.4117	1	0.7439	0.3307	0.3841
t-stat	0.3053	0.7439	1	0.2907	0.3941
BW	0.1450	0.3307	0.2907	1	0.1307
SVMRFE	0.4057	0.3841	0.3941	0.1307	1

Table 11: The overlapping rate between methods on the top 100 genes per class.

Breast Cancer Relevant Genes						Prostate Cancer Relevant Genes					
Gene Symbol	Rank					Gene Symbol	Rank				
	PUG	SNR	t-stat	BW	SVMRFE		PUG	SNR	t-stat	BW	SVMRFE
PIP	1	5745	6146	473	>100	KLK3	4	5	11	61	15
CRABP2	4	5965	6244	498	>100	KLK1	5	3	9	76	16
SCGB2A2	6	6693	6773	458	14	NPY	7	18	22	344	30
CA12	9	6586	6647	518	>100	ACPP	3	4	8	71	12
Uterine Cancer Relevant Genes						Ovarian Cancer Relevant Genes					
Gene Symbol	Rank					Gene Symbol	Rank				
	PUG	SNR	t-stat	BW	SVMRFE		PUG	SNR	t-stat	BW	SVMRFE
ESR1	1	2	16	130	5	OPN	15	334	517	371	63
Hox7	2	4	52	307	12	MMP2	42	2626	3045	481	>100
SCGB2A1	8	3	19	190	4	GPX3	7	411	812	446	>100
PAM	10	83	281	365	71						
HE4	3	1	3	99	5						

Table 12: Detailed comparison between methods on several validated marker genes.

mixture distribution) that is not a straightforward pooled extension of binary (two-class) differential analysis. We emphasize the statistical reproducibility and biological plausibility of the selected gene markers under small sample size, supported by their detailed biological interpretations. We tested our method on six benchmark and in-house real microarray gene expression data sets and compared its performance with that of several existing methods. We imposed a rigorous performance assessment where each and all components of the scheme including gene selection are subjected to cross-validation, for example, held-out/unseen samples in a testing fold are not used for any phase of classifier training.

Tested on six benchmark real microarray data sets, the proposed PUG-OVRSVM method outperforms several widely-adopted gene selection and classification methods with lower error rates, fewer marker genes, and higher performance sustainability. Moreover, while for some data sets, the absolute gain in classification accuracy percentage of PUG-OVRSVM is not dramatically large, it must be recognized that the performance may be approaching the minimum Bayes error rate, in which case PUG-OVRSVM is achieving nearly all the improvement that is theoretically attainable. Furthermore, the improved performance is achieved by correct classifications on some of the most difficult cases, which is considered significant for clinical diagnosis (Ramaswamy et al., 2001). Lastly, although improvements will be data set-dependent, our multi-data set tests have shown that PUG-OVRSVM is the consistent winner as compared to several peer methods.

We note that we have opted for simplicity as opposed to theoretical optimality in designing our method. Our primary goal was to demonstrate that a small number of reproducible phenotypic-dependent genes are sufficient to achieve improved multicategory classification, that is, small sample sizes and a large number of classes need not preclude a high level of performance. Our studies suggest that using genes' marginal associations with the phenotypic categories as we do here has the potential to stabilize the learning process, leading to a substantial reduction in performance variability with small sample size; whereas, the current generation of complex gene selection techniques may not be stable or powerful enough to reliably exploit gene interactions and/or variations unless the sample size is sufficiently large. We have not explored the full flexibility that this method readily allows, with different numbers of subPUGs used by different classifiers. Presumably, equal or better performance could be achieved with fewer genes if more markers were selected for the most difficult classifications, involving the nearest phenotypes. However, such flexibility could actually degrade performance in practice since it introduces extra design choices and, thus, extra sources of variation in classification performance. We may also extend our method to account for variation in fold-changes, with the uncertainty estimated on bootstrap samples judiciously applied to eliminate those PUGs with high variations.

Notably, multicategory classification is intrinsically a nonlinear classification problem, and this method (using one-versus-everyone fold-change based PUG selection, linear kernel SVMs, and the MAP decision rule) is most practically suitable to discriminating unimodal classes. Future work will be required to extend PUG-OVRSVM for multimodal class distributions. An elegant yet simple strategy is to introduce unimodal pseudo-classes for the multi-modal classes via a pre-clustering step, with the final class decision readily made without the need of any decision combiner. Specifically, for each (pseudo-class, super pseudo-class) pair (where, for a pseudo-class originating from class k, the paired super pseudo-class is the union of all pseudo-classes that do not belong to class k), a separating hyperplane is constructed. Accordingly, in selecting subPUGs for each pseudo-class, the pseudo-classes originating from the same class will not be considered.

Acknowledgments

This work was supported in part by the US National Institutes of Health under Grants CA109872, CA149147, CA139246, EB000830, NS29525, and under Contract No. HHSN261200800001E.

References

M. Bakay, Z. Wang, G. Melcon, L. Schiltz, J. Xuan, P. Zhao, V. Sartorelli, J. Seo, E. Pegoraro, C. Angelini, B. Shneiderman, D. Escolar, Y. W. Chen, S. T. Winokur, L. M. Pachman, C. Fan, R. Mandler, Y. Nevo, E. Gordon, Y. Zhu, Y. Dong, Y. Wang, and E. P. Hoffman. Nuclear envelope dystrophies show a transcriptional fingerprint suggesting disruption of Rb-MyoD pathways in muscle regeneration. *Brain*, 129(Pt 4):996–1013, 2006.

U. M. Braga-Neto and E. R. Dougherty. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374–80, 2004.

Z. Cai, R. Goebel, M. R. Salavatipour, and G. Lin. Selecting dissimilar genes for multi-class classification, an application in cancer subtyping. *BMC Bioinformatics*, 8:206, 2007.

R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan, and Y. Wang. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nat Rev Cancer*, 8(1):37–49, 2008.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. Wiley, New York, 2nd edition, 2001.

S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457): 77–87, 2002.

B. P. Durbin, J. S. Hardin, D. M. Hawkins, and D. M. Rocke. A variance-stabilizing transformation for gene-expression microarray data. *Bioinformatics*, 18 Suppl 1:S105–10, 2002.

G. Fort and S. Lambert-Lacroix. Classification using partial least squares with penalized logistic regression. *Bioinformatics*, 21(7):1104–11, 2005.

H. Gish. A probabilistic approach to the understanding and training of neural network classifiers. In *IEEE Intl. Conf. Acoust., Speech, Signal Process.*, pages 1361–1364, 1990.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286 (5439):531–7, 1999.

I. Guyon, J. Weston, S. Barnhill, and V. Vladimir. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

B. Hanczar and E. R. Dougherty. On the comparison of classifiers for microarray data. *Current Bioinformatics*, 5(1):29–39, 2010.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, New York, 2001.

W. Huber, A. von Heydebreck, H. Sultmann, A. Poustka, and M. Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18 Suppl 1:S96–104, 2002.

C. Lai, M. J. Reinders, L. J. van't Veer, and L. F. Wessels. A comparison of univariate and multivariate gene selection techniques for classification of cancer datasets. *BMC Bioinformatics*, 7: 235, 2006.

F. Li and Y. Yang. Analysis of recursive gene selection approaches from microarray data. *Bioinformatics*, 21(19):3741–7, 2005.

T. Li, C. Zhang, and M. Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–37, 2004.

H. Liu, J. Li, and L. Wong. A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, 13:51–60, 2002.

J. J. Liu, G. Cutler, W. Li, Z. Pan, S. Peng, T. Hoey, L. Chen, and X. B. Ling. Multiclass cancer classification and biomarker discovery using GA-based algorithms. *Bioinformatics*, 21(11):2691–7, 2005.

M. Loog, R. P. W. Duin, and R. Haeb-Umbach. Multiclass linear dimension reduction by weighted pairwise fisher criteria. *IEEE Trans Pattern Anal Machine Intell*, 23(7):762–766, 2001.

G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, Hoboken, N.J., 2nd edition, 2008.

C. H. Ooi and P. Tan. Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. *Bioinformatics*, 19(1):37–44, 2003.

S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. Kim, L. C. Goumnerova, P. M. Black, C. Lau, J. C. Allen, D. Zagzag, J. M. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. P. Mesirov, E. S. Lander, and T. R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–42, 2002.

S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov, T. Poggio, W. Gerald, M. Loda, E. S. Lander, and T. R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proc Natl Acad Sci U S A*, 98(26):15149–54, 2001.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2002.

D. M. Rocke and B. Durbin. A model for measurement error for gene expression arrays. *J Comput Biol*, 8(6):557–69, 2001.

K. A. Shedden, J. M. Taylor, T. J. Giordano, R. Kuick, D. E. Misek, G. Rennert, D. R. Schwartz, S. B. Gruber, C. Logsdon, D. Simeone, S. L. Kardia, J. K. Greenson, K. R. Cho, D. G. Beer, E. R. Fearon, and S. Hanash. Accurate molecular classification of human cancers based on gene expression using a simple classifier with a pathological tree-based framework. *Am J Pathol*, 163 (5):1985–95, 2003.

L. Shi, W. D. Jones, R. V. Jensen, S. C. Harris, R. G. Perkins, F. M. Goodsaid, L. Guo, L. J. Croner, C. Boysen, H. Fang, F. Qian, S. Amur, W. Bao, C. C. Barbacioru, V. Bertholet, X. M. Cao, T. M. Chu, P. J. Collins, X. H. Fan, F. W. Frueh, J. C. Fuscoe, X. Guo, J. Han, D. Herman, H. Hong, E. S. Kawasaki, Q. Z. Li, Y. Luo, Y. Ma, N. Mei, R. L. Peterson, R. K. Puri, R. Shippy, Z. Su, Y. A. Sun, H. Sun, B. Thorn, Y. Turpaz, C. Wang, S. J. Wang, J. A. Warrington, J. C. Willey, J. Wu, Q. Xie, L. Zhang, L. Zhang, S. Zhong, R. D. Wolfinger, and W. Tong. The balance of reproducibility, sensitivity, and specificity of lists of differentially expressed genes in microarray studies. *BMC Bioinformatics*, 9 Suppl 9:S10, 2008.

A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 21(5):631–43, 2005.

J. E. Staunton, D. K. Slonim, H. A. Coller, P. Tamayo, M. J. Angelo, J. Park, U. Scherf, J. K. Lee, W. O. Reinhold, J. N. Weinstein, J. P. Mesirov, E. S. Lander, and T. R. Golub. Chemosensitivity prediction by transcriptional profiling. *Proc Natl Acad Sci U S A*, 98(19):10787–92, 2001.

R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proc Natl Acad Sci U S A*, 99(10):6567–72, 2002.

M. E. Tipping and C.M. Bishop. Probabilistic principle component analysis. *Journal of the Royal Statistical Society. Series B*, 61(3):611–622, 1999.

V. N. Vapnik. *Statistical Learning Theory*. Adaptive and learning systems for signal processing, communications, and control. Wiley, New York, 1998.

Y. Wang, J. Lu, R. Lee, Z. Gu, and R. Clarke. Iterative normalization of cNDA microarray data. *IEEE Trans Info. Tech. Biomed*, 6(1):29–37, 2002.

Y. Wang, J. Zhang, J. Khan, R. Clarke, and Z. Gu. Partially-independent component analysis for tissue heterogeneity correction in microarray gene expression analysis. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 24–32, 2003.

Y. Wang, D. J. Miller, and R. Clarke. Approaches to working in high-dimensional data spaces: gene expression microarrays. *Br J Cancer*, 98(6):1023–8, 2008.

Z. Wang, Y. Wang, J. Xuan, Y. Dong, M. Bakay, Y. Feng, R. Clarke, and E. P. Hoffman. Optimized multilayer perceptrons for molecular classification and diagnosis using genomic data. *Bioinformatics*, 22(6):755–61, 2006.

J. Xuan, Y. Wang, Y. Dong, Y. Feng, B. Wang, J. Khan, M. Bakay, Z. Wang, L. Pachman, S. Winokur, Y. W. Chen, R. Clarke, and E. Hoffman. Gene selection for multiclass prediction by weighted fisher criterion. *EURASIP J Bioinform Syst Biol*, page 64628, 2007.

C. H. Yeang, S. Ramaswamy, P. Tamayo, S. Mukherjee, R. M. Rifkin, M. Angelo, M. Reich, E. Lander, J. Mesirov, and T. Golub. Molecular classification of multiple tumor types. *Bioinformatics*, 17 Suppl 1:S316–22, 2001.

J. Zhang, L. Wei, X. Feng, Z. Ma, and Y. Wang. Pattern expression non-negative matrix factorization: Algorithm and application to blind source separation. *Computational Intelligence and Neuroscience*, page Artical ID 168769, 2008.

Y. Zhao, M. C. Li, and R. Simon. An adaptive method for cDNA microarray normalization. *BMC Bioinformatics*, 6:28, 2005.

X. Zhou and D. P. Tuck. MSVM-RFE: extensions of SVM-RFE for multiclass gene selection on DNA microarray data. *Bioinformatics*, 23(9):1106–14, 2007.

Journal of Machine Learning Research 11 (2010) 2169-2173 Submitted 2/10; Revised 8/10; Published 8/10

libDAI: A Free and Open Source C++ Library for Discrete Approximate Inference in Graphical Models

Joris M. Mooij JORIS.MOOIJ@LIBDAI.ORG
Max Planck Institute for Biological Cybernetics
Department for Empirical Inference
Spemannstraße 38, 72076 Tübingen
Germany

Editor: Cheng Soon Ong

Abstract

This paper describes the software package libDAI, a free & open source C++ library that provides implementations of various exact and approximate inference methods for graphical models with discrete-valued variables. libDAI supports directed graphical models (Bayesian networks) as well as undirected ones (Markov random fields and factor graphs). It offers various approximations of the partition sum, marginal probability distributions and maximum probability states. Parameter learning is also supported. A feature comparison with other open source software packages for approximate inference is given. libDAI is licensed under the GPL v2+ license and is available at `http://www.libdai.org`.

Keywords: probabilistic graphical models, approximate inference, open source software, factor graphs, Markov random fields, Bayesian networks

1. Introduction

Probabilistic graphical models (Koller and Friedman, 2009) are at the core of many applications in machine learning nowadays. Because exact inference in graphical models is often infeasible, an extensive literature has evolved in recent years describing various approximation methods for this task. Unfortunately, there exist only few free software implementations for many of these methods. The software package libDAI presented in this paper is the result of an ongoing attempt to improve this situation.

libDAI is a free and open source C++ library (licensed under the GNU General Public License version 2, or higher) that provides implementations of various exact and approximate inference methods for graphical models. libDAI supports factor graphs with discrete variables. This paper describes the most recent libDAI release, version 0.2.7.

The modular design of libDAI makes it easy to implement novel inference algorithms. The large number of inference algorithms already implemented in libDAI allows for easy comparison of accuracy and performance of various algorithms.

2. Inference in Factor Graphs

Although Bayesian networks and Markov random fields are the most commonly used graphical models, libDAI uses a slightly more general type of graphical model: factor graphs (Kschischang

et al., 2001). A *factor graph* is a bipartite graph consisting of *variable* nodes $i \in \mathcal{V}$ and *factor* nodes $I \in \mathcal{F}$, a family of random variables $(X_i)_{i\in\mathcal{V}}$, a family of *factors* $(f_I)_{I\in\mathcal{F}}$ (which are nonnegative functions depending on subsets $N_I \subseteq \mathcal{V}$ of the variables), and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{F}$, where variable node i and factor node I are connected by an undirected edge if and only if the factor f_I depends on variable X_i, that is, if $i \in N_I$. The probability distribution encoded by the factor graph is given by:

$$\mathbb{P}(X_{\mathcal{V}}) = \frac{1}{Z}\prod_{I\in\mathcal{F}} f_I(X_{N_I}), \qquad Z := \sum_{X_{\mathcal{V}}}\prod_{I\in\mathcal{F}} f_I(X_{N_I}).$$

A Bayesian network can easily be represented as a factor graph by introducing a factor for each variable, namely the (conditional) probability table of the variable given its parents in the Bayesian network. A Markov random field can be represented as a factor graph by taking the clique potentials as factors. Factor graphs naturally express the factorization structure of probability distributions. Therefore, they form a convenient representation for approximate inference algorithms that exploit this factorization. This is the reason that libDAI uses factor graphs as the basic representation for graphical models.

Given a factor graph, the following important inference tasks can be distinguished: calculating the *partition sum* Z; calculating *marginal* probability distributions $\mathbb{P}(X_A) = \frac{1}{Z}\sum_{X_{\mathcal{V}\setminus A}}\prod_{I\in\mathcal{F}} f_I(X_I)$ over subsets of variables $X_A, A \subseteq \mathcal{V}$; calculating the *Maximum A Posteriori* (MAP) state, that is, the joint configuration $\operatorname{argmax}_{X_{\mathcal{V}}}\prod_{I\in\mathcal{F}} f_I(X_I)$ which has the maximum probability mass. libDAI offers several inference algorithms which solve these tasks either approximately or exactly, for factor graphs with discrete variables.

2.1 Inference Methods Implemented in libDAI

Apart from exact inference by brute force enumeration and the junction-tree method, libDAI currently offers the following approximate inference methods for calculating partition sums, marginals and MAP states:[1] mean field, (loopy) belief propagation (Kschischang et al., 2001), fractional belief propagation (Wiegerinck and Heskes, 2003), tree-reweighted belief propagation (Wainwright et al., 2003), tree expectation propagation (Minka and Qi, 2004), generalized belief propagation (Yedidia et al., 2005), double-loop generalized belief propagation (Heskes et al., 2003), loop-corrected belief propagation (Mooij and Kappen, 2007; Montanari and Rizzo, 2005), conditioned belief propagation (Eaton and Ghahramani, 2009), a Gibbs sampler and a decimation method.

In addition, libDAI supports parameter learning of conditional probability tables by maximum likelihood or expectation maximization (in case of missing data).

3. Technical Details

libDAI is targeted at researchers that have a good understanding of graphical models. The best way to use libDAI is by writing C++ code that directly invokes the library functions and classes. In addition, part of the functionality is accessible by using various interfaces: a command line interface, a limited MatLab interface, and experimental Python and Octave interfaces (powered by SWIG, see `http://www.swig.org`). Because libDAI is implemented in C++, it is very fast compared with implementations in pure MatLab or Python; the difference in computation time can be up to a few orders of magnitude.

1. Not all inference tasks are implemented by each method.

	libDAI	BNT	PNL	FastInf	GRMM	Factorie	JProGraM
Language	C++	MatLab, C	C++	C++	Java	Scala	Java
License	GPL2+	LGPL2	BSD	GPL3	CPL 1.0	CC-by 3.0	GPL3
Junction tree	+	+	+	+	+	-	+
Belief propagation (BP)	+	+	+	+	+	+	-
Fractional BP	+	-	-	-	-	-	-
Tree-reweighted BP	+	-	-	+	-	-	-
Generalized BP (GBP)	+	-	-	+	+	-	-
Double-loop GBP	+	-	-	-	-	-	-
Loop-corrected BP	+	-	-	-	-	-	-
Conditioned BP	+	-	-	-	-	-	-
Tree expectation propagation	+	-	-	-	-	-	-
Mean field	+	-	-	+	-	-	-
Gibbs sampling	+	+	+	+	+	+	+
Other sampling methods	-	+	+	+	+	+	-
Decimation	+	-	-	-	-	-	-
Continuous variables	-	+	+	-	-	+	+
Dynamic Bayes nets	-	+	+	-	-	+	-
Conditional random fields	-	-	-	-	+	+	-
Relational models	-	-	-	+	-	+	-
Influence diagrams	-	+	-	-	-	-	-
Parameter learning	+	+	+	+	+	+	+
Structure learning	-	+	+	-	-	-	+

Table 1: Feature comparison of various open source software packages for approximate inference on graphical models.

libDAI is designed to be cross-platform and is known to compile on GNU/Linux distributions and on Mac OS X using the `GCC` compiler suite, and also under Windows using either `CygWin` or `MS Visual Studio 2008`.

The libDAI sources and documentation can be downloaded or browsed from the libDAI website at `http://www.libdai.org`. The Google group libDAI (see `http://groups.google.com/group/libdai`) can be used for getting support and discussing development issues. Extensive documentation generated by Doxygen (see `http://www.doxygen.org`) is available, both as part of the source code and online. Unit tests and various example programs are provided, including the full source code of the solver which won in two categories of the *2010 UAI Approximate Inference Challenge* (see also `http://www.cs.huji.ac.il/project/UAI10/`) and uses several of the algorithms implemented in libDAI.

4. Related Work

Other prominent open source software packages supporting both directed and undirected graphical models are the Bayes Net Toolbox (BNT, see `http://bnt.googlecode.com`), the Probabilistic Networks Library (PNL, see `http://sourceforge.net/projects/openpnl`), FastInf (Jaimovich et al., 2010), GRMM (`http://mallet.cs.umass.edu/grmm`), Factorie (`http://code.google.com/p/factorie`), and JProGraM (`http://jprogram.sourceforge.net`). A feature comparison of libDAI with these other packages is provided in Table 1.

Acknowledgments

Part of this work was part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant BSIK03024. I would like to express my gratitude to the following people who made major contributions to libDAI: Martijn Leisink (for laying down the foundations for the library), Frederik Eaton (for contributing the Gibbs sampler, conditioned BP, fractional BP and various other improvements), Charles Vaske (for contributing the parameter learning algorithms), Giuseppe Passino (for various improvements), Bastian Wemmenhove (for contributing the MR algorithm), Patrick Pletscher (for contributing the SWIG interface). I am also indebted to the following people for bug fixes and miscellaneous smaller patches: Claudio Lima, Christian Wojek, Sebastian Nowozin, Stefano Pellegrini, Ofer Meshi, Dan Preston, Peter Gober, Jiuxiang Hu, Peter Rockett, Dhruv Batra, Alexander Schwing, Alejandro Lage, Matt Dunham.

References

F. Eaton and Z. Ghahramani. Choosing a variable to clamp. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, volume 5 of *JMLR Workshop and Conference Proceedings*, pages 145–152, 2009.

T. Heskes, K. Albers, and B. Kappen. Approximate inference and constrained optimization. In C. Meek and U. Kjærulff, editors, *Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 313–320. Morgan Kaufmann, 2003.

A. Jaimovich, O. Meshi, I. McGraw, and G. Elidan. FastInf: An efficient approximate inference library. *Journal of Machine Learning Research*, 11:17331736, 2010.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

F. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

T. Minka and Y. Qi. Tree-structured approximations by expectation propagation. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 193–200. MIT Press, 2004.

A. Montanari and T. Rizzo. How to compute loop corrections to the Bethe approximation. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(10):P10011, 2005.

J. Mooij and B. Kappen. Loop corrections for approximate inference on factor graphs. *Journal of Machine Learning Research*, 8:1113–1143, 2007.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In C. Bishop and B. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (AISTATS 2003)*, 2003.

W. Wiegerinck and T. Heskes. Fractional belief propagation. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 438–445. MIT Press, 2003.

J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

Journal of Machine Learning Research 11 (2010) 2175-2198 Submitted 7/07; Revised 2/09; Published 8/10

Learning Gradients: Predictive Models that Infer Geometry and Statistical Dependence

Qiang Wu WUQIANG@MATH.MSU.EDU
Department of Mathematics
Michigan State University
East Lansing, MI 48824, USA

Justin Guinney JUSTIN.GUINNEY@SAGEBASE.ORG
Sage Bionetworks
Fred Hutchinson Cancer Research Center
Seattle, WA 98109, USA

Mauro Maggioni[*] MAURO.MAGGIONI@DUKE.EDU
Sayan Mukherjee[†] SAYAN@STAT.DUKE.EDU
Departments of Mathematics and Statistical Science
Duke University
Durham, NC 27708, USA

Editor: Marina Meila

Abstract

The problems of dimension reduction and inference of statistical dependence are addressed by the modeling framework of learning gradients. The models we propose hold for Euclidean spaces as well as the manifold setting. The central quantity in this approach is an estimate of the gradient of the regression or classification function. Two quadratic forms are constructed from gradient estimates: the gradient outer product and gradient based diffusion maps. The first quantity can be used for supervised dimension reduction on manifolds as well as inference of a graphical model encoding dependencies that are predictive of a response variable. The second quantity can be used for nonlinear projections that incorporate both the geometric structure of the manifold as well as variation of the response variable on the manifold. We relate the gradient outer product to standard statistical quantities such as covariances and provide a simple and precise comparison of a variety of supervised dimensionality reduction methods. We provide rates of convergence for both inference of informative directions as well as inference of a graphical model of variable dependencies.

Keywords: gradient estimates, manifold learning, graphical models, inverse regression, dimension reduction, gradient diffusion maps

1. Introduction

The problem of developing predictive models given data from high-dimensional physical and biological systems is central to many fields such as computational biology. A premise in modeling natural phenomena of this type is that data generated by measuring thousands of variables lie on or near a low-dimensional manifold. This hearkens to the central idea of reducing data to only relevant

∗. Also in the Department of Computer Science.
†. Also in the Department of Computer Science and Institute for Genome Sciences & Policy.

information. This idea was fundamental to the paradigm of Fisher (1922) and goes back at least to Adcock (1878) and Edgeworth (1884). For an excellent review of this program see Cook (2007). In this paper we examine how this paradigm can be used to infer geometry of the data as well as statistical dependencies relevant to prediction.

The modern reprise of this program has been developed in the broad areas of manifold learning and simultaneous dimension reduction and regression. Manifold learning has focused on the problem of projecting high-dimensional data onto a few directions or dimensions while respecting local structure and distances. A variety of unsupervised methods have been proposed for this problem (Tenenbaum et al., 2000; Roweis and Saul, 2000; Belkin and Niyogi, 2003; Donoho and Grimes, 2003). Simultaneous dimension reduction and regression considers the problem of finding directions that are informative with respect to predicting the response variable. These methods can be summarized by three categories: (1) methods based on inverse regression (Li, 1991; Cook and Weisberg, 1991; Fukumizu et al., 2005; Wu et al., 2007), (2) methods based on gradients of the regression function (Xia et al., 2002; Mukherjee and Zhou, 2006; Mukherjee and Wu, 2006), (3) methods based on combining local classifiers (Hastie and Tibshirani, 1996; Sugiyama, 2007). Our focus is on the supervised problem. However, we will use the core idea in manifold learning, local estimation.

We first illustrate with a simple example how gradient information can be used for supervised dimension reduction. Both linear projections as well as nonlinear embeddings based on gradient estimates are used for supervised dimension reduction. In both approaches the importance of using the response variable is highlighted.

The main contributions in this paper consist of (1) development of gradient based diffusion maps, (2) precise statistical relations between the above three categories of supervised dimension reduction methods, (3) inference of graphical models or conditional independence structure given gradient estimates, (4) rates of convergence of the estimated graphical model. The rate of convergence depends on a geometric quantity, the intrinsic dimension of the gradient on the manifold supporting the data, rather than the sparsity of the graph.

2. A Statistical Foundation for Learning Gradients

The problem of regression can be summarized as estimating the function

$$f_r(x) = \mathbb{E}(Y \mid X = x)$$

from data $D = \{L_i = (Y_i, X_i)\}_{i=1}^n$ where X_i is a vector in a p-dimensional compact metric space $X \in \mathcal{X} \subset \mathbb{R}^p$ and $Y_i \in \mathbb{R}$ is a real valued output. Typically the data are drawn iid from a joint distribution, $L_i \overset{iid}{\sim} \rho(X,Y)$ thus specifying a model

$$y_i = f_r(x_i) + \varepsilon_i$$

with ε_i drawn iid from a specified noise model. We define ρ_X as the marginal distribution of the explanatory variables.

In this paper we will explain how inference of the gradient of the regression function

$$\nabla f_r = \left(\frac{\partial f_r}{\partial x^1}, ..., \frac{\partial f_r}{\partial x^p} \right)^T$$

provides information on the geometry and statistical dependencies relevant to predicting the response variable given the explanatory variables. Our work is motivated by the following ideas. The gradient is a local concept as it measures local changes of a function. Integrating information encoded by the gradient allows for inference of the geometric structure in the data relevant to the response. We will explore two approaches to integrate this local information. The first approach is averaging local gradient estimates and motivates the study of the gradient outer product (GOP) matrix. The GOP can be used to motivate a variety of linear supervised dimension reduction formulations. The GOP can also be considered as a covariance matrix and used for inference of conditional dependence between predictive explanatory variables. The second approach is to paste local gradient estimates together. This motivates the study of gradient based diffusion maps (GDM). This operator can be used for nonlinear supervised dimension reduction by embedding the support of the marginal distribution onto a much lower dimensional manifold that varies with respect to the response.

The gradient outer product Γ is a $p \times p$ matrix with elements

$$\Gamma_{ij} = \left\langle \frac{\partial f_r}{\partial x^i}, \frac{\partial f_r}{\partial x^j} \right\rangle_{L^2_{\rho_X}},$$

where ρ_X is the marginal distribution of the explanatory variables and $L^2_{\rho_X}$ is the space of square integrable functions with respect to the measure ρ_X. Using the notation $a \otimes b = ab^T$ for $a, b \in \mathbb{R}^p$, we can write

$$\Gamma = \mathbb{E}(\nabla f_r \otimes \nabla f_r).$$

This matrix provides global information about the predictive geometry of the data (developed in Section 2.2) as well as inference of conditional independence between variables (developed in Section 5). The GOP is meaningful in both the Euclidean as well as the manifold setting where the marginal distribution ρ_X is concentrated on a much lower dimensional manifold $\mathcal{M}$ of dimension $d_{\mathcal{M}}$ (developed in Section 4.1.2).

Since the GOP is a global quantity and is constructed by averaging the gradient over the marginal distribution of the data it cannot isolate local information or local geometry. This global summary of the joint distribution is advantageous in statistical analyses where global inferences are desired: global predictive factors comprised of the explanatory variables or global estimates of statistical dependence between explanatory variables. It is problematic to use a global summary for constructing a nonlinear projection or embedding that captures the local predictive geometry on the marginal distribution.

Random walks or diffusions on manifolds and graphs have been used for a variety of nonlinear dimension reduction or manifold embedding procedures (Belkin and Niyogi, 2003, 2004; Szummer and Jaakkola, 2001; Coifman et al., 2005a,b). Our basic idea is to use local gradient information to construct a random walk on a graph or manifold based on the ideas of diffusion analysis and diffusion geometry (Coifman and Lafon, 2006; Coifman and Maggioni, 2006). The central quantity in diffusion based approaches is the definition of a diffusion operator L based on a similarity metric W_{ij} between two points x_i and x_j. A commonly used diffusion operator is the graph Laplacian

$$L = I - D^{-1/2} W D^{-1/2}, \quad \text{where } D_{ii} = \sum_j W_{ij}.$$

Dimension reduction is achieved by projection onto a spectral decomposition of the operator L or powers of the operator L^t which corresponds to running the diffusion for some time t.

We propose the following similarity metric called the gradient based diffusion map (GDM) to smoothly paste together local gradient estimates

$$W_{ij} = W_f(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_1} - \frac{|\frac{1}{2}(\nabla f_r(x_i) + \nabla f_r(x_j)) \cdot (x_i - x_j)|^2}{\sigma_2} \right).$$

In the above equation the first term $\|x_i - x_j\|^2$ encodes the local geometry of the marginal distribution and the second term pastes together gradient estimates between neighboring points. The first term is used in unsupervised dimension reduction methods such as Laplacian eigenmaps or diffusion maps (Belkin and Niyogi, 2003). The second term can be interpreted as a diffusion map on the function values, the approximation is due to a first order Taylor expansion

$$\frac{1}{2}(\nabla f_r(x_i) + \nabla f_r(x_j)) \cdot (x_i - x_j) \approx f_r(x_i) - f_r(x_j) \qquad \text{if } x_i \approx x_j.$$

A related similarity was briefly mentioned in Coifman et al. (2005b) and used in Szlam et al. (2008) in the context of semi-supervised learning. In Section 4.2 we study this nonlinear projection method under the assumption that the marginal distribution is concentrated on a much lower dimensional manifold $\mathcal{M}$.

2.1 Illustration of Linear Projections and Nonlinear Embeddings

The simple example in this section fixes the differences between linear projections and nonlinear embeddings using either diffusion maps or gradient based diffusion maps. The marginal distribution is uniform on the following manifold

$$X_1 = t\cos(t), \quad X_2 = 70h, \quad X_3 = t\sin(t) \text{ where } t = \frac{3\pi}{2}(1 + 2\theta), \theta \in [0,1], h \in [0,1],$$

and the regression function is $Y = \sin(5\pi\theta)$, see Figure 1(a). In this example a two dimensional manifold is embedded in $\mathbb{R}^3$ and the variation of the response variable can be embedded onto one dimension. The points in Figure 1(a) are the points on the manifold and the false color signifies the function value at these points. In Figure 1(b) the data is embedded in two dimensions using diffusion maps with the function values displayed in false color. It is clear that the direction of greatest variation is X_2 which corresponds to h. It is not the direction along which the regression function has greatest variation. In Figure 1(c) the data is projected onto two axes using the GOP approach and we see that the relevant dimensions X_1, X_3 are recovered. This example also shows that linear dimension reduction may still make sense in the manifold setting. In Figure 1(d) the data is embedded using gradient based diffusion maps and we capture the direction θ in which the data varies with respect to the regression function.

2.2 Gradient Outer Product Matrix and Dimension Reduction

In the case of linear supervised dimension reduction we assume the following semi-parametric model holds

$$Y = f_r(X) + \varepsilon = g(b_1^T X, \ldots, b_d^T X) + \varepsilon, \tag{1}$$

where ε is noise and $B = (b_1^T, ..., b_d^T)$ is the dimension reduction (DR) space. In this case the number of explanatory variables p is large but the the response variable Y depends on a few directions in $\mathbb{R}^p$

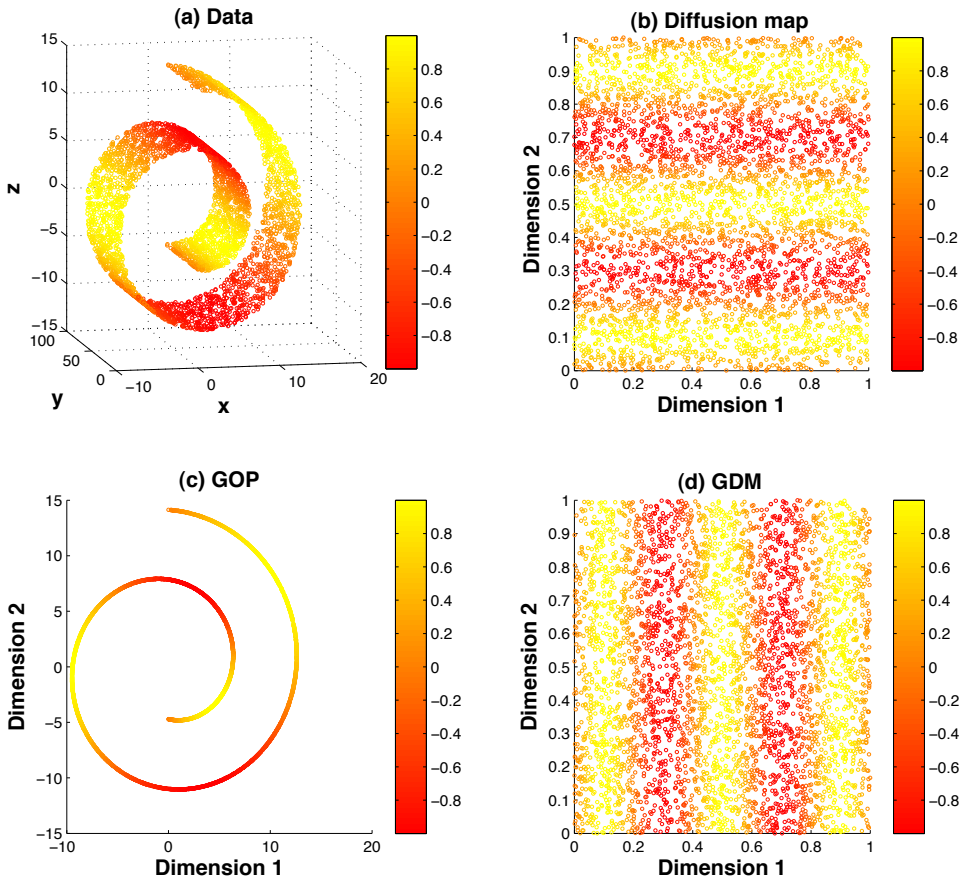


Figure 1: (a) Plot of the original three dimensional data with the color of the point corresponding to the function value; (b) Embedding the data onto two dimensions using diffusion maps, here dimensions 1 and 2 corresponds to h and θ respectively; (c) Projection of the data onto two dimensions using gradient based dimension reduction, here dimensions 1 and 2 corresponds to x and z respectively; (d) Embedding the data onto two dimensions using gradient based diffusion maps, here dimensions 1 and 2 corresponds to θ and h respectively.

and dimension reduction becomes the central problem in finding an accurate regression model. In the following we develop theory relating the gradient of the regression function to the above model of dimension reduction.

Observe that for a vector $v \in \mathbb{R}^p$, $\frac{\partial f_r(x)}{\partial v} = v \cdot \nabla f_r$ is identically zero if f_r does not depends on v and is not zero if f_r changes along the direction v. The following lemma relates the gradient outer product matrix to supervised dimension reduction.

Lemma 1 *Under the assumptions of the semi-parametric model* (1)*, the gradient outer product matrix* Γ *is of rank at most* d*. Denote by* $\{v_1, \ldots, v_d\}$ *the eigenvectors associated to the nonzero eigenvalues of* Γ*. The following holds*

$$\operatorname{span}(B) = \operatorname{span}(v_1, \ldots, v_d).$$

Lemma 1 states the dimension reduction space can be computed by a spectral decomposition of Γ. Notice that this method does not require additional geometric conditions on the distribution. This is in contrast to other supervised dimension reduction methods (Li, 1991; Cook and Weisberg, 1991; Li, 1992) that require a geometric or distributional requirement on X, namely that the level sets of the distribution are elliptical.

This observation motivates supervised dimension reduction methods based on consistent estimators Γ_D of Γ given data D. Several methods have been motivated by this idea, either implicitly as in minimum average variance estimation (MAVE) (Xia et al., 2002), or explicitly as in outer product of gradient (OPG) (Xia et al., 2002) and learning gradients (Mukherjee et al., 2010).

The gradient outer product matrix is defined globally and its relation to dimension reduction in Section 2.2 is based on global properties. However, since the gradient itself is a local concept we can also study the geometric structure encoded in the gradient outer product matrix from a local point of view.

2.3 Gradient Outer Product Matrix as a Covariance Matrix

A central concept used in dimension reduction is the covariance matrix of the inverse regression function $\Omega_{X|Y} = \operatorname{cov}_Y[\mathbb{E}_X(X \mid Y)]$. The fact that $\Omega_{X|Y}$ encodes the DR directions $B = (b_1, \ldots, b_d)$ under certain conditions was explored in Li (1991).

The main result of this subsection is to relate the two matrices: Γ and $\Omega_{X|Y}$. The first observation from this relation is that the gradient outer product matrix is a covariance matrix with a very particular construction. The second observation is that the gradient outer product matrix contains more information than the covariance of the inverse regression since it captures local information. This is outlined for linear regression and then generalized to nonlinear regression. Proofs of the propositions and the underlying mathematical ideas will be developed in Section 4.1.1.

The linear regression problem is often stated as

$$Y = \beta^T X + \varepsilon, \quad \mathbb{E}\,\varepsilon = 0. \tag{2}$$

For this model the following relation between gradient estimates and the inverse regression holds.

Proposition 2 *Suppose* (2) *holds. Given the covariance of the inverse regression,* $\Omega_{X|Y} = \operatorname{cov}_Y(\mathbb{E}_X(X \mid Y))$*, the variance of the output variable,* $\sigma_Y^2 = \operatorname{var}(Y)$*, and the covariance of the input variables,* $\Sigma_X = \operatorname{cov}(X)$*, the gradient outer product matrix is*

$$\Gamma = \sigma_Y^2 \left(1 - \frac{\sigma_\varepsilon^2}{\sigma_Y^2}\right)^2 \Sigma_X^{-1} \Omega_{X|Y} \Sigma_X^{-1}, \tag{3}$$

assuming that Σ_X *is full rank.*

The above result states that for a linear model the matrices Γ and $\Omega_{X|Y}$ are equivalent modulo a scale parameter—approximately the variance of the output variable—and a rotation—the precision matrix (inverse of the covariance matrix) of the input variables.

In order to generalize Proposition 2 to the nonlinear regression setting we first consider piecewise linear functions. Suppose there exists a non-overlapping partition of the input space

$$\mathcal{X} = \bigcup_{i=1}^{I} R_i$$

such that in each region R_i the regression function f_r is linear and the noise has zero mean

$$f_r(x) = \beta_i^T x, \quad \mathbb{E}\,\varepsilon_i = 0 \quad \text{for } x \in R_i. \tag{4}$$

The following corollary is true.

Corollary 3 *Given partitions R_i of the input space for which* (4) *holds, define in each partition R_i the following local quantities: the covariance of the input variables $\Sigma_i = \mathrm{cov}(X \in R_i)$, the covariance of the inverse regression $\Omega_i = \mathrm{cov}_Y(\mathbb{E}_X(X \in R_i \mid Y))$, the variance of the output variable $\sigma_i^2 = \mathrm{var}(Y \mid X \in R_i)$. Assuming that matrices Σ_i are full rank, the gradient outer product matrix can be computed in terms of these local quantities*

$$\Gamma = \sum_{i=1}^{I} \rho_X(R_i)\sigma_i^2 \left(1 - \frac{\sigma_{\varepsilon_i}^2}{\sigma_i^2}\right)^2 \Sigma_i^{-1}\,\Omega_i\,\Sigma_i^{-1}, \tag{5}$$

where $\rho_X(R_i)$ is the measure of partition R_i with respect to the marginal distribution ρ_X.

If the regression function is smooth it can be approximated by a first order Taylor series expansion in each partition R_i provided the region is small enough. Theoretically there always exist partitions such that the locally linear models (4) hold approximately (i.e., $\mathbb{E}\,\varepsilon_i \approx 0$). Therefore (5) holds approximately

$$\Gamma \approx \sum_{i=1}^{I} \rho_X(R_i)\sigma_i^2 \left(1 - \frac{\sigma_{\varepsilon_i}^2}{\sigma_i^2}\right)^2 \Sigma_i^{-1}\,\Omega_i\,\Sigma_i^{-1}.$$

Supervised dimension reduction methods based on the covariance of the inverse regression require an elliptic condition on X. This condition ensures that $\Omega_{X|Y}$ encodes the DR subspace but not necessarily the entire DR subspace. In the worst case it is possible that $\mathbb{E}(X \mid Y) = 0$, $\Omega_{X|Y} = \mathbf{0}$. and as a result $\Omega_{X|Y}$ contains no information. In the linear case $\Omega_{X|Y}$ will encode the full DR subspace, the one predictive direction. Since the GOP is an average the inverse covariance matrix of the locally linear models it contains all the predictive directions. This motivates the centrality of the gradient outer product.

This derivation of the gradient outer product matrix based on local variation has two potential implications. It provides a theoretical comparison between dimension reduction approaches based on the gradient outer product matrix and inverse regression. This will be explored in Section 4.1.1 in detail. The integration of local variation will be used to infer statistical dependence between the explanatory variables conditioned on the response variable in Section 5.

A common belief in high dimensional data analysis is that the data are concentrated on a low dimensional manifold. Both theoretical and empirical evidence of this belief is accumulating. In the manifold setting, the input space is a manifold $\mathcal{X} = \mathcal{M}$ of dimension $d_{\mathcal{M}} \ll p$. We assume the existence of an isometric embedding $\varphi : \mathcal{M} \to \mathbb{R}^p$ and the observed input variables $(x_i)_{i=1}^n$ are the image of points $(q_i)_{i=1}^n$ drawn from a distribution on the manifold: $x_i = \varphi(q_i)$. In this case, global

statistics are not as meaningful from a modeling perspective.[1] In this setting the gradient outer product matrix should be defined in terms of the gradient on the manifold, $\nabla_{\mathcal{M}} f_r$,

$$\Gamma = \mathbb{E}\left(d\varphi(\nabla_{\mathcal{M}} f_r) \otimes d\varphi(\nabla_{\mathcal{M}} f_r)\right) = \mathbb{E}\left(d\varphi(\nabla_{\mathcal{M}} f_r \otimes \nabla_{\mathcal{M}} f_r)(d\varphi)^T\right).$$

This quantity is meaningful from a modeling perspective because gradients on the manifold capture the local structure in the data. Note that the $d_{\mathcal{M}} \times d_{\mathcal{M}}$ matrix $\Gamma_{\mathcal{M}} = \nabla_{\mathcal{M}} f_r \otimes \nabla_{\mathcal{M}} f_r$ is the central quantity of interest in this setting. However, we know neither the manifold nor the coordinates on the manifold and are only provided points in the ambient space. For this reason we cannot compute $\Gamma_{\mathcal{M}}$. However, we can understand its properties by analyzing the gradient outer product matrix Γ in the ambient space, a $p \times p$ matrix. Details on conditions under which Γ provides information on $\Gamma_{\mathcal{M}}$ are developed in Section 4.1.2.

3. Estimating Gradients

An estimate of the gradient is required in order to estimate the gradient outer product matrix Γ. Many approaches for computing gradients exist including various numerical derivative algorithms, local linear/polynomial smoothing (Fan and Gijbels, 1996), and learning gradients by kernel models (Mukherjee and Zhou, 2006; Mukherjee and Wu, 2006). Our main focus is on what can be done given an estimate of Γ rather than estimation methods for the gradient. The application domain we focus on is the analysis of high-dimensional data with few observations, where $p \gg n$ and some traditional methods do not work well because of computational complexity or numerical stability. Learning gradients by kernel models was specifically developed for this type of data in the Euclidean setting for regression (Mukherjee and Zhou, 2006) and classification (Mukherjee and Wu, 2006). The same algorithms were shown to be valid for the manifold setting with a different interpretation in Mukherjee et al. (2010). In this section we review the formulation of the algorithms and state properties that will be relevant in subsequent sections.

The motivation for learning gradients is based on Taylor expanding the regression function

$$f_r(u) \approx f_r(x) + \nabla f_r(x) \cdot (u - x), \text{ for } x \approx u,$$

which can be evaluated at data points $(x_i)_{i=1}^n$

$$f_r(x_i) \approx f_r(x_j) + \nabla f_r(x_j) \cdot (x_i - x_j), \text{ for } x_i \approx x_j.$$

Given data $D = \{(y_i, x_i)\}_{i=1}^n$ the objective is to simultaneously estimate the regression function f_r by a function f_D and the gradient ∇f_r by the p-dimensional vector valued function $\vec{f}_D$.

In the regression setting the following regularized loss functional provides the estimates (Mukherjee and Zhou, 2006).

Definition 4 *Given the data $D = \{(x_i, y_i)\}_{i=1}^n$, define the first order difference error of function f and vector-valued function $\vec{f} = (f_1, \ldots, f_p)$ on D as*

$$\mathcal{E}_D(f, \vec{f}) = \frac{1}{n^2} \sum_{i,j=1}^{n} w_{i,j}^s \left(y_i - f(x_j) + \vec{f}(x_i) \cdot (x_j - x_i) \right)^2.$$

1. Consider $\Omega(X \mid Y)$ as an example. For any given y, the set $\{x \mid Y = y\}$ is a submanifold. The global mean will not necessarily lie on the manifold. Therefore, from a modeling perspective $\mathbb{E}(X \mid Y)$ and hence $\Omega(X \mid Y)$ may convey nothing about the manifold. Although this does not mean it is useless in practice, a theoretical justification seems impossible.

The regression function and gradient estimate is modeled by

$$(f_D, \vec{f}_D) := \arg \min_{(f,\vec{f}) \in \mathcal{H}_K^{p+1}} \left(\mathcal{E}_D(f,\vec{f}) + \lambda_1 \|f\|_K^2 + \lambda_2 \|\vec{f}\|_K^2 \right),$$

where f_D and $\vec{f}_D$ are estimates of f_r and ∇f_r given the data, $w_{i,j}^s$ is a weight function with bandwidth s, $\|\cdot\|_K$ is the reproducing kernel Hilbert space (RKHS) norm, λ_1 and λ_2 are positive constants called the regularization parameters, the RKHS norm of a p-vector valued function is the sum of the RKHS norm of its components $\|\vec{f}\|_K^2 := \sum_{t=1}^p \|\vec{f}_t\|_K^2$.

A typical weight function is a Gaussian $w_{i,j}^s = \exp(-\|x_i - x_j\|^2/2s^2)$. Note this definition is slightly different from that given in Mukherjee and Zhou (2006) where $f(x_j)$ is replaced by y_j and only the gradient estimate $\vec{f}_D$ is estimated.

In the classification setting we are given $D = \{(y_i, x_i)\}_{i=1}^n$ where $y_i \in \{-1, 1\}$ are labels. The central quantity here is the classification function which we can define by conditional probabilities

$$f_c(x) = \log \left[\frac{\rho(Y=1 \mid x)}{\rho(Y=-1 \mid x)} \right] = \arg\min \mathbb{E}\, \phi(Y f(X))$$

where $\phi(t) = \log(1 + e^{-t})$ and the sign of f_c is a Bayes optimal classifier. The following regularized loss functional provides estimates for the classification function and gradient (Mukherjee and Wu, 2006).

Definition 5 *Given a sample $D = \{(x_i, y_i)\}_{i=1}^n$ we define the empirical error as*

$$\mathcal{E}_D^\phi(f,\vec{f}) = \frac{1}{n^2} \sum_{i,j=1}^n w_{ij}^s \phi\Big(y_i \big(f(x_j) + \vec{f}(x_i) \cdot (x_i - x_j) \big) \Big).$$

The classification function and gradient estimate given a sample is modeled by

$$(f_D, \vec{f}_D) = \arg \min_{(f,\vec{f}) \in \mathcal{H}_K^{p+1}} \left(\mathcal{E}_D^\phi(f,\vec{f}) + \lambda_1 \|f\|_K^2 + \lambda_2 \|\vec{f}\|_K^2 \right),$$

where f_D and $\vec{f}_D$ are estimates of f_c and ∇f_c, and λ_1, λ_2 are the regularization parameters.

In the manifold setting the above algorithms are still valid. However the interpretation changes. We state the regression case, the classification case is analogous (Mukherjee et al., 2010).

Definition 6 *Let $\mathcal{M}$ be a Riemannian manifold and $\varphi : \mathcal{M} \to \mathbb{R}^p$ be an isometric embedding which is unknown. Denote $\mathcal{X} = \varphi(\mathcal{M})$ and $\mathcal{H}_K = \mathcal{H}_K(\mathcal{X})$. For the sample $D = \{(q_i, y_i)\}_{i=1}^n \in (\mathcal{M} \times \mathbb{R})^n$, $x_i = \varphi(q_i) \in \mathbb{R}^p$, the learning gradients algorithm on $\mathcal{M}$ provides estimates*

$$(f_D, \vec{f}_D) := \arg \min_{f,\vec{f} \in \mathcal{H}_K^{p+1}} \left\{ \frac{1}{n^2} \sum_{i,j=1}^n w_{i,j}^s \left(y_i - f(x_j) + \vec{f}(x_i) \cdot (x_j - x_i) \right)^2 + \lambda_1 \|f\| + \lambda_2 \|\vec{f}\|_K^2 \right\},$$

where $\vec{f}_D$ is a model for $\mathrm{d}\varphi(\nabla_{\mathcal{M}} f_r)$ and f_D is a model for f_r.

From a computational perspective the advantage of the RKHS framework is that in both regression and classification the solutions satisfy a representer theorem (Wahba, 1990; Mukherjee and Zhou, 2006; Mukherjee and Wu, 2006)

$$f_D(x) = \sum_{i=1}^{n} \alpha_{i,D} K(x, x_i), \;\; \vec{f}_D(x) = \sum_{i=1}^{n} c_{i,D} K(x, x_i), \tag{6}$$

with $c_D = (c_{1,D}, \ldots, c_{n,D}) \in \mathbb{R}^{p \times n}$, and $\alpha_D = (\alpha_{1,D}, ..., \alpha_{n,D})^T \in \mathbb{R}^p$. In Mukherjee and Zhou (2006) and Mukherjee and Wu (2006) methods for efficiently computing the minima were introduced in the setting where $p \gg n$. The methods involve linear systems of equations of dimension nd where $d \leq n$.

The consistency of the gradient estimates for both regression and classification were proven in Mukherjee and Zhou (2006) and Mukherjee and Wu (2006) respectively.

Proposition 7 *Under mild conditions (see Mukherjee and Zhou, 2006; Mukherjee and Wu, 2006 for details) the estimates of the gradients of the regression or classification function f converge to the true gradients: with probability greater than $1 - \delta$,*

$$\|\vec{f}_D - \nabla f\|_{L^2_{\rho_x}} \leq C \log\left(\frac{2}{\delta}\right) n^{-1/p}.$$

Consistency in the manifold setting was studied in Mukherjee et al. (2010) and the rate of convergence was determined by the dimension of the manifold, $d_{\mathcal{M}}$, not the dimension of the ambient space p.

Proposition 8 *Under mild conditions (see Mukherjee et al., 2010 for details), with probability greater than $1 - \delta$,*

$$||(d\varphi)^* \vec{f}_D - \nabla_{\mathcal{M}} f||_{L^2_{\rho_{\mathcal{M}}}} \leq C \log\left(\frac{2}{\delta}\right) n^{-1/d_{\mathcal{M}}},$$

where where $(d\varphi)^*$ is the dual of the map $d\varphi$.

4. Dimension Reduction Using Gradient Estimates

In this section we study some properties of dimension reduction using the gradient estimates. We also relate learning gradients to previous approaches for dimension reduction in regression.

4.1 Linear Dimension Reduction

The theoretical foundation for linear dimension reduction using the spectral decomposition of the gradient outer production matrix was developed in Section 2.2. The estimate of the gradient obtained by the kernel models in Section 3 provides the following empirical estimate of the gradient outer product matrix

$$\hat{\Gamma} := c_D K^2 c_D^T = \frac{1}{n} \sum_{i=1}^{n} \vec{f}_D(x_i) \otimes \vec{f}_D(x_i),$$

where K is the kernel matrix with $K_{ij} = K(x_i, x_j)$ and c_D is defined in (6). The eigenvectors corresponding to the top eigenvalues of $\hat{\Gamma}$ can be used to estimate the d dimension reduction directions. The following proposition states that the estimate is consistent.

Proposition 9 *Suppose that f satisfies the semi-parametric model* (1) *and $\vec{f}_D$ is an empirical approximation of ∇f. Let $\hat{v}_1, \ldots, \hat{v}_d$ be the eigenvectors of $\hat{\Gamma}$ associated to the top d eigenvalues. The following holds*

$$\lim_{n\to\infty} \text{span}(\hat{v}_1, \ldots, \hat{v}_d) = \text{span}(B).$$

Moreover, the remaining eigenvectors correspond to eigenvalues close to 0.

Proof : Proposition 7 implies that $\lim_{n\to\infty} \hat{\Gamma}_{ij} = \Gamma_{ij}$ and hence $\lim_{n\to\infty} \Gamma = \Gamma$ in matrix norm. By perturbation theory the eigenvalues (see Golub and Loan, 1996, Theorem 8.1.4 and Corollary 8.1.6) and eigenvectors (see Zwald and Blanchard, 2006) of $\hat{\Gamma}$ converge to those of Γ respectively. The conclusions then follow from Lemma 1. ■

Proposition 9 justifies linear dimension reduction using consistent gradient estimates from a global point of view.

In the next subsection we study the gradient outer product matrix from the local point of view and provide details on the relation between gradient based methods and sliced inverse regression.

4.1.1 RELATION TO SLICED INVERSE REGRESSION (SIR)

The SIR method computes the DR directions using a generalized eigen-decomposition problem

$$\Omega_{X|Y}\beta = \nu\Sigma_X\beta. \tag{7}$$

In order to study the relation between our method with SIR, we study the relation between the matrices $\Omega_{X|Y}$ and Γ.

We start with a simple model where the DR space contains only one direction which means the regression function satisfies the following semi-parametric model

$$Y = g(\beta^T X) + \varepsilon$$

where $\|\beta\| = 1$ and $\mathbb{E}\,\varepsilon = 0$. The following theorem holds and Proposition 2 is a special case.

Theorem 10 *Suppose that Σ_X is invertible. There exists a constant C such that*

$$\Gamma = C\Sigma_X^{-1}\Omega_{X|Y}\Sigma_X^{-1}.$$

If g is a linear function the constant is $C = \sigma_Y^2\left(1 - \frac{\sigma_\varepsilon^2}{\sigma_Y^2}\right)^2$.

Proof It is proven in Duan and Li (1991) that

$$\Omega_{X|Y} = \text{var}(h(Y))\Sigma_X\beta\beta^T\Sigma_X$$

where $h(y) = \frac{\mathbb{E}(\beta^T(X-\mu)|y)}{\beta^T\Sigma_X\beta}$ with $\mu = \mathbb{E}(X)$ and Σ_X is the covariance matrix of X. In this case, the computation of matrix Γ is direct:

$$\Gamma = \mathbb{E}[(g'(\beta^T X))^2]\beta\beta^T.$$

By the assumption Σ_X is invertible, we immediately obtain the first relation with

$$C = \mathbb{E}[(g'(\beta^T X))^2]\text{var}(h(Y))^{-1}.$$

If $g(t) = at + b$, we have $h(y) = \frac{y-b-\beta^T\mu}{a\beta^T\Sigma_X\beta}$ and consequently

$$\mathrm{var}(h(Y)) = \frac{\sigma_Y^2}{a^2(\beta^T\Sigma_X\beta)^2}.$$

By the simple fact $\mathbb{E}(g'(\beta^T X)^2] = a^2$ and $\sigma_Y^2 = a^2\beta^T\Sigma_X\beta + \sigma_\varepsilon^2$, we get

$$C = \frac{a^4(\beta^T\Sigma_X\beta)^2}{\sigma_Y^2} = \frac{(\sigma_Y^2 - \sigma_\varepsilon^2)^2}{\sigma_Y^2} = \sigma_Y^2\left(1 - \frac{\sigma_\varepsilon^2}{\sigma_Y^2}\right)^2.$$

This finishes the proof. ■

It is apparent that Γ and $\Omega_{X|Y}$ differ only up to a linear transformation. As a consequence the generalized eigen-decomposition (7) of $\Omega_{X|Y}$ with respect to Σ_X yields the same first direction as the eigen-decomposition of Γ.

Consider the linear case. Without loss of generality suppose X is normalized to satisfy $\Sigma_X = \sigma^2 I$, we see $\Omega_{X|Y}$ is the same as Γ up to a constant of about $\frac{\sigma_Y^2}{\sigma^4}$. Notice that this factor measures the ratio of the variation of the response to the variation over the input space as well as along the predictive direction. This implies that Γ is more informative because it not only contains the information of the descriptive directions but also measures their importance with respect to the change of the response variable.

When there are more than one DR directions as in model (1), we partition the input space into I small regions $X = \bigcup_{i=1}^I R_i$ such that over each region R_i the response variable y is approximately linear with respect to x and the descriptive direction is a linear combination of the column vectors of B. By the discussion in Section 2.2

$$\Gamma = \sum_i \rho_X(R_i)\,\Gamma_i \approx \sum_{i=1}^I \rho_X(R_i)\sigma_i^2\,\Sigma_i^{-1}\,\Omega_i\Sigma_i^{-1},$$

where Γ_i is the gradient outer product matrix on R_i and $\Omega_i = \mathrm{cov}_Y(\mathbb{E}_X(X \in R_i \mid Y))$. In this sense, the gradient covariance matrix Γ can be regarded as the weighted sum of the local covariance matrices of the inverse regression function. Recall that SIR suffers from the possible degeneracy of the covariance matrix of the inverse regression function over the entire input space while the local covariance matrix of the inverse regression function will not be degenerate unless the function is constant. Moreover, in the gradient outer product matrix, the importance of local descriptive directions are also taken into account. These observations partially explain the generality and some advantages of gradient based methods.

Note this theoretical comparison is independent of the method used to estimate the gradient. Hence the same comparison holds between SIR and other gradient based methods such as mean average variance estimation (MAVE) and outer product of gradients (OPG) developed in Xia et al. (2002).

4.1.2 THEORETICAL FEASIBILITY OF LINEAR PROJECTIONS FOR NONLINEAR MANIFOLDS

In this section we explore why linear projections based on the gradient outer product matrix are feasible and have meaning when the manifold structure is nonlinear. The crux of the analysis will be demonstrating that the estimated gradient outer product matrix $\hat{\Gamma}$ is still meaningful.

Again assume there exists an unknown isometric embedding of the manifold onto the ambient space, $\varphi : \mathcal{M} \to \mathbb{R}^p$. From a modeling perspective we would like the gradient estimate from data $\vec{f}_D$ to approximate $d\varphi(\nabla_{\mathcal{M}} f_r)$ (Mukherjee et al., 2010). Generally this is not true when the manifold is nonlinear, φ is a nonlinear map. Instead, the estimate provides the following information about $\nabla_{\mathcal{M}} f_r$

$$\lim_{n\to\infty} (d\varphi)^* \vec{f}_D = \nabla_{\mathcal{M}} f_r,$$

where $(d\varphi)^*$ is the dual of $d\varphi$, the differential of φ.

Note that f_r is not well defined on any open set of $\mathbb{R}^p$. Hence, it is not meaningful to consider the gradient of ∇f_r in the ambient space $\mathbb{R}^p$. Also, we cannot recover directly the gradient of f_r on the manifold since we know neither the manifold nor the embedding. However, we can still recover the DR directions from the matrix $\hat{\Gamma}$.

Assume f_r satisfies the semi-parametric model (1). The matrix Γ is not well defined but $\hat{\Gamma}$ is well defined. The following proposition ensures that the spectral decomposition of $\hat{\Gamma}$ provides the DR directions.

Proposition 11 *If $v \perp b_i$ for all $i = 1, \ldots, d$, then* $\lim_{n\to\infty} v^T \hat{\Gamma} v = 0$.

Proof Let $\vec{f}_\lambda$ be the sample limit of $\vec{f}_D$, that is

$$\vec{f}_\lambda = \arg\min_{\vec{f}\in\mathcal{H}_K^p} \left\{ \int_{\mathcal{M}} \int_{\mathcal{M}} e^{-\frac{\|x-\xi\|^2}{2s^2}} \left(f_r(x) - f_r(\xi) + \vec{f}(x)\cdot(\xi - x) \right)^2 d\rho_{\mathcal{M}}(x) d\rho_{\mathcal{M}}(\xi) + \lambda \|\vec{f}\|_K^2 \right\}.$$

By the assumption and a simple rotation argument we can show that $v \cdot \vec{f}_\lambda = 0$.

It was proven in Mukherjee and Zhou (2006) that $\lim_{n\to\infty} \|\vec{f}_D - \vec{f}_\lambda\|_K = 0$. A result of this is for $\hat{\Xi} = c_D K c_D^T$

$$v^T \hat{\Xi} v = \|v \cdot \vec{f}_D\|_K^2 \overset{n\to\infty}{\longrightarrow} \|v \cdot \vec{f}_\lambda\|_K^2 = 0.$$

This implies that $\lim_{n\to\infty} v^T \hat{\Gamma} v = 0$ and proves the proposition. ■

Proposition 11 states that all the vectors perpendicular to the DR space correspond to eigenvalues near zero of $\hat{\Gamma}$ and will be filtered out. This means the DR directions can be still found by the spectral decomposition of the estimated gradient outer product matrix.

4.2 Nonlinear Projections: Gradient Based Diffusion Maps (GDM)

As discussed in Section 2 the gradient of the regression function can be used for nonlinear projections. The basic idea was to use local gradient information to construct a diffusion operator L based on a similarity metric W_{ij} between two points x_i and x_j. A commonly used diffusion operator is the graph Laplacian

$$L = I - D^{-1/2} W D^{-1/2}, \quad \text{where } D_{ii} = \sum_j W_{ij}.$$

Dimension reduction is achieved by projection onto a spectral decomposition of the operator L or powers $(I-L)^t$ of the operator $(I-L)$ which corresponds to running the diffusion $(I-L)$ for some time t. The gradient based diffusion map (GDM) was defined as

$$W_{ij} = W_f(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_1} - \frac{|\frac{1}{2}(\nabla f_r(x_i) + \nabla f_r(x_j)) \cdot (x_i - x_j)|^2}{\sigma_2} \right). \tag{8}$$

In the above equation the first term $\|x_i - x_j\|^2$ encodes the local geometry of the marginal distribution and the second term pastes together gradient estimates between neighboring points. The first term is used in unsupervised dimension reduction methods such as Laplacian eigenmaps or diffusion maps (Belkin and Niyogi, 2003). The second term can be interpreted as a first order Taylor expansion leading to the following approximation

$$\frac{1}{2}(\nabla f_r(x_i) + \nabla f_r(x_j)) \cdot (x_i - x_j) \approx f_r(x_i) - f_r(x_j).$$

The form (8) is closely related to the following function adapted similarity proposed in Szlam et al. (2008)

$$W_f(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_1} - \frac{|f(x_i) - f(x_j)|^2}{\sigma_2} \right),$$

where the function evaluations $f(x_i)$ are computed based on a first rough estimate of the regression function from the data.

The utility of nonlinear dimension reduction has been shown to be dramatic with respect to prediction accuracy in the semi-supervised learning setting where a large set of unlabeled data, $\{x_1, ..., x_u\}$, drawn from the marginal distribution, were used to learn the projection and a small set of labeled data $\{(y_1, x_1), ..., (y_\ell, x_\ell)\}$ were used to learn the regression function on the projected data. Of practical importance in this setting is the need to evaluate the similarity function on out of sample data. The labeled data is used to compute the gradient estimate, which can be evaluated on out-of-sample data. Given the gradient estimate and the labeled and unlabeled data $(x_1, ..., x_\ell, x_{\ell+1}, x_{\ell+u})$ the following GDM can be defined on all the samples

$$\tilde{W}_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_1} - \frac{|\frac{1}{2}(\vec{f}_D(x_i) + \vec{f}_D(x_j)) \cdot (x_i - x_j)|^2}{\sigma_2} \right), \quad i, j = 1, .., u + \ell.$$

An analysis of the accuracy of the GDM approach is based on how well $f(x_i) - f(x_j)$ can be estimated using the gradient estimate $\vec{f}_D$. The first order Taylor expansion on the manifold results in the following approximation

$$f(x_i) - f(x_j) \approx \nabla_{\mathcal{M}} f(x_i) \cdot v_{ij}, \text{ for } v_{ij} \approx 0,$$

where $v_{ij} \in T_{x_i}\mathcal{M}$ is the tangent vector such that $x_j = \mathrm{Exp}_{x_i}(v_{ij})$ where Exp_{x_i} is the exponential map at x_i (see do Carmo, 1992; Mukherjee et al., 2010). Since we cannot compute $\nabla_{\mathcal{M}} f$ we use $\vec{f}_D$. The following proposition states that estimates of the function value differences can be accurately estimated from gradient estimate $\vec{f}_D$.

Proposition 12 *The following holds*

$$f_r(x_i) - f_r(x_j) \approx \vec{f}_D(x_i) \cdot (x_i - x_j), \textit{ for } x_i \approx x_j.$$

Proof By the fact $x_i - x_j \approx \mathrm{d}\varphi(v_{ij})$ we have

$$\vec{f}_D(x_i) \cdot (x_i - x_j) \approx \langle \vec{f}_D(x_i), \mathrm{d}\varphi(v_{ij}) \rangle = \langle (\mathrm{d}\varphi)^*(\vec{f}_D(x_i)), v_{ij} \rangle \approx \langle \nabla_{\mathcal{M}} f_r(x_i), v_{ij} \rangle$$

which implies the conclusion. ■

This proposition does not prove consistency of the GDM approach. This research program of proving convergence of the eigenvectors of a graph Laplacian to the eigenfunctions of a corresponding Laplace-Beltrami operator is a source of extensive research in diffusion maps (Belkin and Niyogi, 2005; Giné and Koltchinskii, 2006). It would be of interest to adapt these approaches to the gradient setting. The limiting operator will in general depend on how σ_1 and σ_2 approach 0 as the number of points tends to infinity. For example, it is easy to see that if $\sigma_1, \sigma_2 \to 0_+$ suitably as $n \to +\infty$, with $\sigma_1/\sigma_2 = \alpha$, then the limiting operator is the Laplacian on the manifold $(x, \alpha f(x)) \subset \mathcal{M} \times \mathbb{R}$.

4.2.1 EMPIRICAL RESULTS FOR GRADIENT BASED DIFFUSION MAPS

In this section we motivate the efficacy of the GDM approach with an empirical study of predictive accuracy in the semi-supervised setting on six benchmark data sets found in Chapelle et al. (2006). In the semi-supervised setting using the labeled as well as the unlabeled data for dimension reduction followed by fitting a regression model has often increased predictive accuracy. We used the benchmark data so we could compare the performance of DM and GDM to eleven algorithms (Chapelle et al., 2006, Table 21.11). The conclusion of our study is that GDM improves predictive accuracy over DM and that GDM is competitive with respect to the other algorithms.

For each data set twelve splits were generated with 100 samples labeled in each split. We applied DM and GDM to each of these sets to find DR directions. We projected the data (labeled and unlabeled) onto the DR directions and used a k-Nearest-Neighbor (kNN) classifier to classify the unlabeled data. The parameters of the DM, GDM, and number of neighbors were set using a validation set in each trial. The average classification error rate for the unlabeled data over the twelve splits are reported in Table 1. We also report in Table 1 the top performing algorithm for the data sets in Chapelle et al. (2006, Table 21.11). Laplacian RLS stands for Laplacian regularized least-squares, SGT stands for spectral graph transducer, Cluster-Kernel is an algorithm that uses two kernel functions, see (Chapelle et al., 2006, Chapter 11) for details.

A reasonable conclusion from Table 1 is that having label information improves the performance of diffusion operator with respect to prediction. In addition, dimension reduction using GDM followed by a simple classifier is competitive to other approaches. We suspect that integrating GDM with a penalized classification algorithm in the same spirit as Laplacian regularized least-squares can improve performance.

5. Graphical Models and Conditional Independence

One example of a statistical analysis where global inferences are desired or explanations with respect to the coordinates of the data is important is a graphical model over undirected graphs. In this setting it is of interest to understand how coordinates covary with respect to variation in response, as is provided by the GOP. Often of greater interest is to infer direct or conditional dependencies between two coordinates as a function of variation in the response. In this section we explore how this can be done using the GOP.

A natural idea in multivariate analysis is to model the conditional independence of a multivariate distribution using a graphical model over undirected graphs. The theory of Gauss-Markov graphs

Data	DM	GDM	Best
G241C	19.96%	18.61%	13.49% (Cluster-Kernel)
G241D	14.27%	13.64%	4.95% (Cluster-Kernel)
Digit1	1.8%	1.8%	1.8% (DM, GDM)
BCI	48.53%	31.36%	31.36% (GDM, Laplacian RLS)
USPS	12.85%	10.76%	4.68% (Laplacian RLS)
Text	24.71%	23.57%	23.09% (SGT)

Table 1: Error rates for DM and GDM over six data sets reported in Chapelle et al. (2006, Table 21.11). The column 'Best' reports the error rate for the algorithm with the smallest error of the 13 applied to the data.

(Speed and Kiiveri, 1986; Lauritzen, 1996) was developed for multivariate Gaussian densities

$$p(x) \propto \exp\left(-\frac{1}{2}x^T Jx + h^T x\right),$$

where the covariance is J^{-1} and the mean is $\mu = J^{-1}h$. The result of the theory is that the precision matrix J, given by $J = \Sigma_X^{-1}$, provides a measurement of conditional independence. The meaning of this dependence is highlighted by the partial correlation matrix R_X where each element R_{ij} is a measure of dependence between variables i and j conditioned on all other variables $S^{/ij}$ and $i \neq j$

$$R_{ij} = \frac{\text{cov}(X_i, X_j \mid S^{/ij})}{\sqrt{\text{var}(X_i \mid S^{/ij})}\sqrt{\text{var}(X_j \mid S^{/ij})}}.$$

The partial correlation matrix is typically computed from the precision matrix J

$$R_{ij} = -J_{ij}/\sqrt{J_{ii}J_{jj}}.$$

In the regression and classification framework inference of the conditional dependence between explanatory variables has limited information. Much more useful would be the conditional dependence of the explanatory variables conditioned on variation in the response variable. In Section 2 we stated that both the covariance of the inverse regression as well as the gradient outer product matrix provide estimates of the covariance of the explanatory variables conditioned on variation in the response variable. Given this observation, the inverses of these matrices

$$J_{X|Y} = \Omega_{X|Y}^{-1} \quad \text{and} \quad J_\Gamma = \Gamma^{-1},$$

provide evidence for the conditional dependence between explanatory variables conditioned on the response. We focus on the inverse of the gradient outer product matrix in this paper since it is of use for both linear and nonlinear functions.

The two main approaches to inferring graphical models in high-dimensional regression have been based on either sparse factor models (Carvalho et al., 2008) or sparse graphical models representing sparse partial correlations (Meinshausen and Buhlmann, 2006). Our approach differs from both of these approaches in that the response variable is always explicit. For sparse factor models

the factors can be estimated independent of the response variable and in the sparse graphical model the response variable is considered as just another node, the same as the explanatory variables. Our approach and the sparse factor models approach both share an assumption of sparsity in the number of factors or directions. Sparse graphical model approaches assume sparsity of the partial correlation matrix.

Our proof of the convergence of the estimated conditional dependence matrix $(\hat{\Gamma})^{-1}$ to the population conditional dependence matrix Γ^{-1} relies on the assumption that the gradient outer product matrix being low rank. This again highlights the difference between our modeling assumption of low rank versus sparsity of the conditional dependence matrix. Since we assume that both Γ and $\hat{\Gamma}$ are singular and low rank we use pseudo-inverses in order to construct the dependence graph.

Proposition 13 *Let Γ^{-1} be the pseudo-inverse of Γ. Let the eigenvalues and eigenvectors of $\hat{\Gamma}$ be $\hat{\lambda}_i$ and $\hat{v}_i$ respectively. If $\varepsilon > 0$ is chosen so that $\varepsilon = \varepsilon_n = o(1)$ and $\varepsilon_n^{-1}\|\hat{\Gamma} - \Gamma\| = o(1)$, then the convergence*

$$\sum_{\hat{\lambda}_i > \varepsilon} \hat{v}_i \hat{\lambda}_i^{-1} \hat{v}_i \stackrel{n\to\infty}{\longrightarrow} \Gamma^{-1}$$

holds in probability.

Proof We have shown in Proposition 9 that $\|\hat{\Gamma} - \Gamma\| = o(1)$. Denote the eigenvalues and eigenvectors of Γ as λ_i and v_i respectively. Then

$$|\hat{\lambda}_i - \lambda_i| = O(\|\hat{\Gamma} - \Gamma\|) \qquad \text{and} \qquad \|\hat{v}_i - v_i\| = O(\|\hat{\Gamma} - \Gamma\|).$$

By the condition $\varepsilon_n^{-1}\|\hat{\Gamma} - \Gamma\| = o(1)$ the following holds

$$\hat{\lambda}_i > \varepsilon \Longrightarrow \lambda_i > \varepsilon/2 \Longrightarrow \lambda_i > 0$$

implying $\{i : \hat{\lambda}_i > \varepsilon\} \subset \{i : \lambda_i > 0\}$ in probability. On the other hand, denoting $\tau = \min\{\lambda_i : \lambda_i > 0\}$, the condition $\varepsilon_n = o(1)$ implies

$$\{i : \lambda_i > 0\} = \{i : \lambda \geq \tau\} \subset \{i : \hat{\lambda}_i \geq \tau/2\} \subset \{i : \hat{\lambda}_i > \varepsilon\}$$

in probability. Hence we obtain

$$\{i : \lambda_i > 0\} = \{i : \hat{\lambda}_i > \varepsilon\}$$

in probability.

For each $j \in \{i : \lambda_i > 0\}$ we have $\lambda_j, \hat{\lambda}_j \geq \tau/2$ in probability, so

$$|\hat{\lambda}_j^{-1} - \lambda_j^{-1}| \leq |\hat{\lambda}_j - \lambda_j|/(2\tau) \stackrel{n\to\infty}{\longrightarrow} 0.$$

Thus we finally obtain

$$\sum_{\hat{\lambda}_i > \varepsilon} \hat{v}_i \hat{\lambda}_i^{-1} \hat{v}_i \stackrel{n\to\infty}{\longrightarrow} \sum_{\lambda_i > 0} v_i \lambda_i^{-1} v_i^T = \Gamma^{-1}.$$

This proves the conclusion. ■

5.1 Results on Simulated and Real Data

We first provide an intuition of the ideas behind our inference of graphical models using simple simulated data. We then apply the method to study dependencies in gene expression in the development of prostate cancer.

5.1.1 SIMULATED DATA

The following simple example clarifies the information contained in the covariance matrix as well as the gradient outer product matrix. Construct the following dependent explanatory variables from standard random normal variables $\theta_1, ..., \theta_5 \overset{iid}{\sim} \mathcal{N}(0,1)$

$$X_1 = \theta_1, X_2 = \theta_1 + \theta_2, X_3 = \theta_3 + \theta_4, X_4 = \theta_4, X_5 = \theta_5 - \theta_4,$$

and the following response

$$Y = X_1 + (X_3 + X_5)/2 + \varepsilon_1,$$

where $\varepsilon_1 \sim \mathcal{N}(0, .5^2)$.

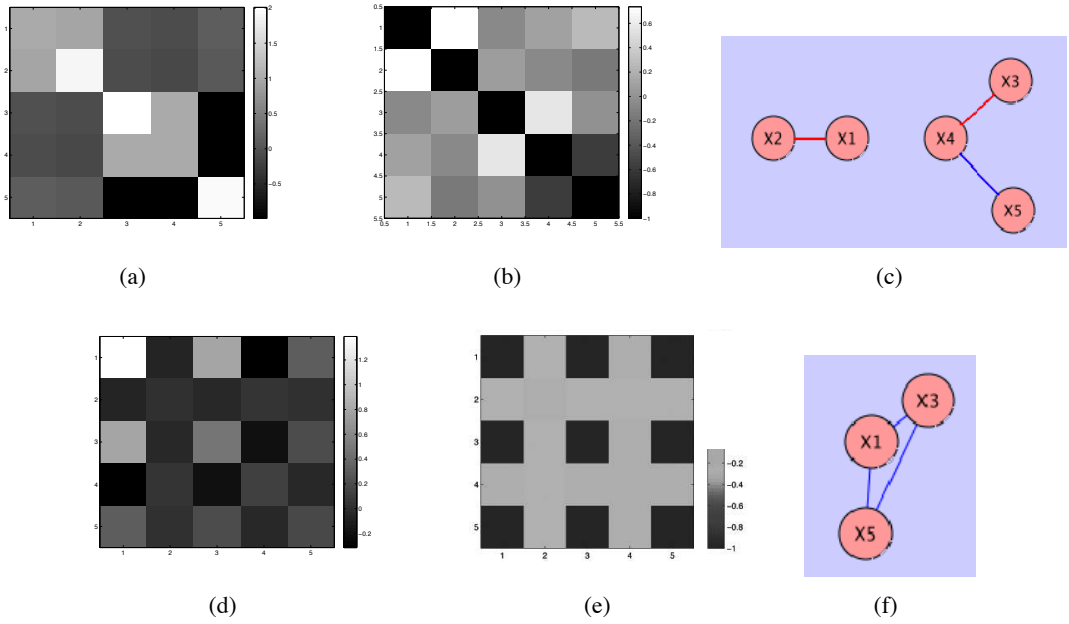
We drew 100 observations $(x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}, y_i)_{i=1}^{100}$ from the above sampling design. From this data we estimate the covariance matrix of the marginals $\hat{\Sigma}_X$ and the gradient outer product matrix $\hat{\Gamma}$. From $\hat{\Sigma}_X$, Figure 2(a), we see that X_1 and X_2 covary with each other and X_3, X_4, X_5 covary. The conditional independence matrix $\hat{R}_X$, Figure 2(b), provides information on more direct relations between the coordinates as we see that X_5 is independent of X_3 given X_4, $X_5 \perp\!\!\!\perp X_3 \mid X_4$. The dependence relations are summarized in the graphical model in Figure 2(c). Taking the response variable into account, we find in the gradient outer product matrix, Figure 2(d), the variables X_2 and X_4 are irrelevant while X_1, X_3, X_5 are relevant. The matrix $\hat{R}_{\hat{\Gamma}}$ is shown in Figure 2(e) and implies that any pair of X_1, X_3, X_5 are negatively dependent conditioned on the other and the response variable Y. The graphical model is given in Figure 2(f).

5.1.2 GENES DRIVING PROGRESSION OF PROSTATE CANCER

A fundamental problem in cancer biology is to understand the molecular and genetic basis of the progression of a tumor from less serious states to more serious states. An example is the progression from a benign growth to malignant cancer. The key interest in this problem is to understand the genetic basis of cancer. A classic model for the genetic basis of cancer was proposed by Fearon and Vogelstein (1990) describing a series of genetic events that cause progression of colorectal cancer.

In Edelman et al. (2008) the inverse of the gradient outer product was used to infer the dependence between genes that drive tumor progression in prostate cancer and melanoma. In the case of melanoma the data consisted of genomewide expression data from normal, primary, and metastatic skin samples. Part of the analysis in this paper was inference of conditional dependence graphs or networks of genes that drive differential expression between stages of progression. The gradient outer product matrix was used to infer detailed models of gene networks that may drive tumor progression.

In this paper, we model gene networks relevant in driving progression in prostate cancer as an illustration of how the methodology can be used to posit biological hypotheses. The objective is to understand the dependence structure between genes that are predictive of progression from benign to malignant prostate cancer. in progressing from benign to malignant prostate cancer. The data consists of 22 benign and 32 advanced prostate tumor samples (Tomlins et al., 2007; Edelman et al.,



(a) (b) (c)

(d) (e) (f)

Figure 2: (a) Covariance matrix $\hat{\Sigma}_X$; (b) Partial correlation matrix $\hat{R}_X$; (c) Graphical model representation of partial correlation matrix; (d) Gradient outer product matrix $\hat{\Gamma}$; (e) Partial correlations $\hat{R}_{\hat{\Gamma}}$ with respect to $\hat{\Gamma}$; (f) Graphical model representation of $\hat{R}_{\hat{\Gamma}}$.

2008). For each sample the expression level of over 12,000 probes corresponding to genes were measured. We eliminated many of those probes with low variation across all samples resulting in a 4095 probes or variables. From this reduced data set we estimated the gradient outer product matrix, $\hat{\Gamma}$, and used the pseudo-inverse to compute the conditional independence matrix, $\hat{J} = (\hat{\Gamma})^{-1}$. From the conditional independence matrix we computed the partial correlation matrix $\hat{R}$ where $\hat{R}_{ij} = -\frac{\hat{J}_{ij}}{\sqrt{\hat{J}_{ii}\hat{J}_{jj}}}$ for $i \neq j$ and 0 otherwise. We again reduced the R matrix to obtain 139 nodes and 400 edges corresponding to the largest partial correlations and construct the graph seen in Figure 3.

The structure of the partial correlation graph recapitulates some known biological processes in the progression of prostate cancer. The most highly connected gene is MME (labeled green) which is known to have significant deregulation in prostate cancer and is associated with aggressive tumors (Tomlins et al., 2007). We also observe two distinct clusters annotated in yellow and purple in the graph that we call C_1 and C_2 respectively. These clusters derive their associations principally through 5 genes, annotated in light blue and dark blue in the graph. The light blue genes AMACR, ANXA1, and CD38 seem to have strong dependence with respect to the genes in C_1 wile C_2 is dependent on these genes in addition to the dark blue genes LMAN1L and SLC14A1. AMACR and ANXA1 as well as CD38 are well-known to have roles in prostate cancer progression (Jiang et al., 2004; Hsiang et al., 2004; Kramer et al., 1995). The other two genes LMAN1L and SLC14A1 are known to have tumorigenic properties and would be candidates for further experiments to better understand their role in prostate cancer.

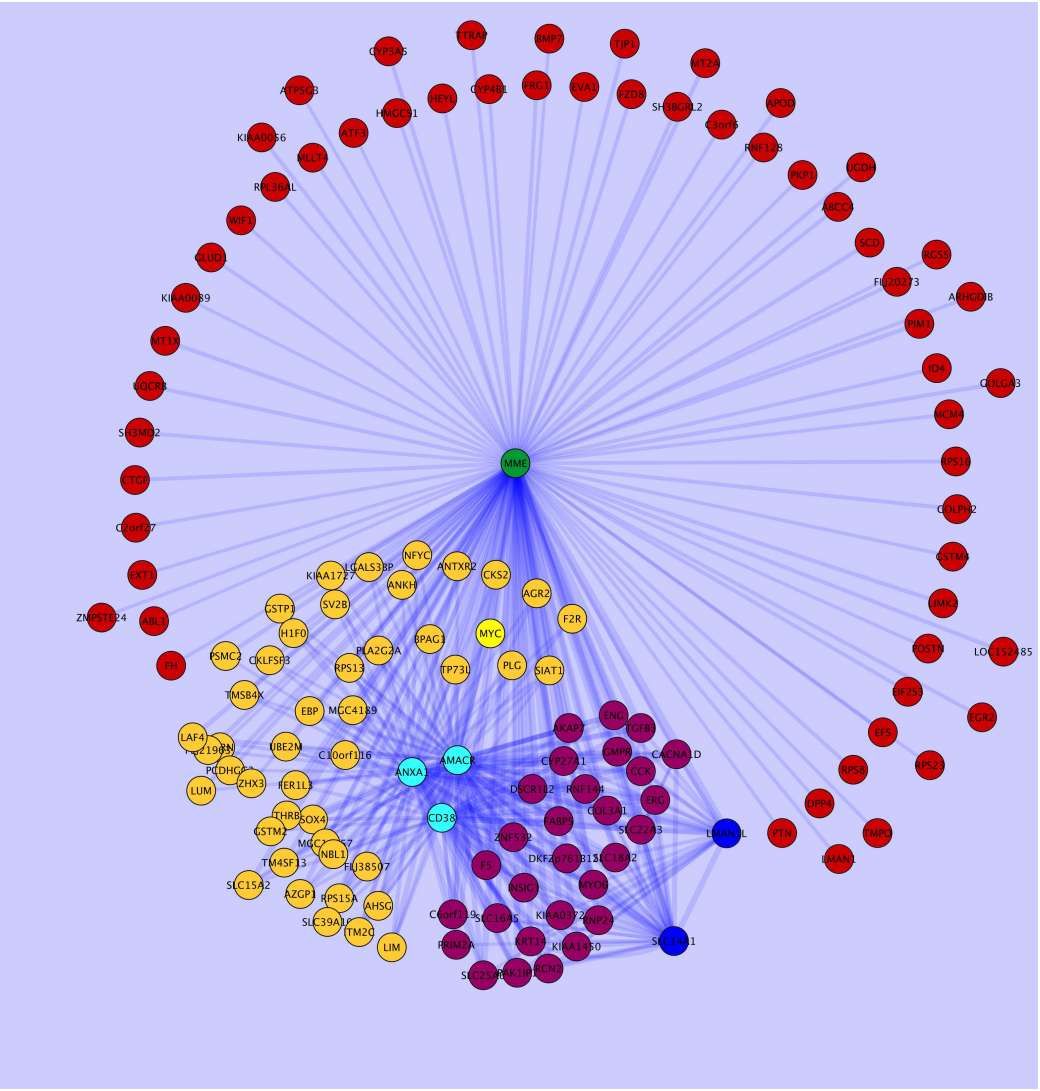


Figure 3: Graphical model of genes relevant in tumors progressing from benign to malignant prostate tissue. The edges correspond to partial correlations.

6. Discussion

The main contribution of this paper is to describe how inference of the gradient of the regression or classification function encodes information about the predictive geometry as well as the predictive conditional dependence in the data. Two methods are introduced gradient based diffusion maps and inference of conditional independence structure given gradient estimates. Precise statistical relations between different approaches to supervised dimension reduction are described. Simulated and real data are used to illustrate the utility of the methods developed. We prove convergence of the estimated graphical model to the population dependence graph. We find this direct link be-

tween graphical models and dimension reduction intriguing and suggest that the manifold learning perspective holds potential in the analysis and inference of graphical models.

Acknowledgments

This work was partially supported by NSF grant DMS-0732260, NIH Systems Biology Center Grant, and NIH R01 CA123175-01A1. MM is grateful for partial support from NSF (DMS 0650413, IIS 0803293), ONR N00014-07-1-0625, the Sloan Foundation and Duke.

References

R.J. Adcock. A problem in least squares. *The Analyst*, 5:53–54, 1878.

M. Belkin and P. Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data representation. *Neural Computation*, 15(6):1373–1396, 2003.

M. Belkin and P. Niyogi. Semi-Supervised Learning on Riemannian Manifolds. *Machine Learning*, 56(1-3):209–239, 2004.

M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. In *Learning Theory*, volume 3559 of *Lecture Notes in Comput. Sci.*, pages 486–500. Springer, Berlin, 2005.

C. Carvalho, J. Lucas, Q. Wang, J. Chang, J. Nevins, and M. West. High-dimensional sparse factor modelling - applications in gene expression genomics. *Journal of the American Statistical Association*, 103(484):1438–1456, 2008.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`.

R.R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21 (1):5–30, 2006.

R.R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, and S.W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431, 2005a.

R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, and S.W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Multiscale methods. *Proceedings of the National Academy of Sciences*, 102(21):7432–7437, 2005b.

R.D. Cook. Fisher lecture: Dimension reduction in regression. *Statistical Science*, 22(1):1–26, 2007.

R.D. Cook and S. Weisberg. Discussion of "Sliced inverse regression for dimension reduction". *J. Amer. Statist. Assoc.*, 86:328–332, 1991.

M. P. do Carmo. *Riemannian Geometry*. Birkhäuser, Boston, MA, 1992.

D. Donoho and C. Grimes. Hessian eigenmaps: new locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100:5591–5596, 2003.

N. Duan and K.C. Li. Slicing regression: a link-free regression method. *Ann. Statist.*, 19(2):505–530, 1991.

E.J. Edelman, J. Guinney, J-T. Chi, P.G. Febbo, and S. Mukherjee. Modeling cancer progression via pathway dependencies. *PLoS Comput. Biol.*, 4(2):e28, 2008.

F.Y. Edgeworth. On the reduction of observations. *Philosophical Magazine*, pages 135–141, 1884.

J. Fan and I. Gijbels. *Local Polynomial Modelling and its Applications*. Chapman and Hall, London, 1996.

E.R. Fearon and B. Vogelstein. A genetic model for colorectal tumorigenesis. *Cell*, 61:759–767, 1990.

R.A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Statistical Society A*, 222:309–368, 1922.

K. Fukumizu, F.R. Bach, and M.I. Jordan. Dimensionality reduction in supervised learning with reproducing kernel Hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2005.

E. Giné and V. Koltchinskii. Empirical graph Laplacian approximation of Laplace-Beltrami operators: large sample results. In *High dimensional probability*, volume 51 of *IMS Lecture Notes Monogr. Ser.*, pages 238–259. Inst. Math. Statist., Beachwood, OH, 2006.

G.H. Golub and C.F. Va Loan. *Matrix Computations*. The Johns Hopkins University Press; 3rd edition, 1996.

T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *J. Roy.Statist. Soc. Ser. B*, 58(1):155–176, 1996.

C-H. Hsiang, T. Tunoda, Y.E. Whang, D.R. Tyson, and D.K. Ornstein. The impact of altered annexin i protein levels on apoptosis and signal transduction pathways in prostate cancer cells. *The Prostate*, 66(13):1413–1424, 2004.

Z. Jiang, B.A. Woda BA, C.L. Wu, and X.J. Yang. Discovery and clinical application of a novel prostate cancer marker: alpha-methylacyl CoA racemase (P504S). *Am. J. Clin. Pathol*, 122(2): 275–8941, 2004.

G . Kramer, G. Steiner, D. Fodinger, E. Fiebiger, C. Rappersberger, S. Binder, J. Hofbauer, and M. Marberger. High expression of a CD38-like molecule in normal prostatic epithelium and its differential loss in benign and malignant disease. *The Journal of Urology*, 154(5):1636–1641, 1995.

S.L. Lauritzen. *Graphical Models*. Oxford: Clarendon Press, 1996.

K.C. Li. Sliced inverse regression for dimension reduction. *J. Amer. Statist. Assoc.*, 86:316–342, 1991.

K.C. Li. On principal Hessian directions for data visualization and dimension reduction: another application of Stein's lemma. *Ann. Statist.*, 97:1025–1039, 1992.

N. Meinshausen and P. Buhlmann. High-dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34(2):1436–1462, 2006.

S. Mukherjee and Q. Wu. Estimation of gradients and coordinate covariation in classification. *J. Mach. Learn. Res.*, 7:2481–2514, 2006.

S. Mukherjee and DX. Zhou. Learning coordinate covariances via gradients. *J. Mach. Learn. Res.*, 7:519–549, 2006.

S. Mukherjee, D-X. Zhou, and Q. Wu. Learning gradients and feature selection on manifolds. *Bernoulli*, 16(1):181–207, 2010.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

T. Speed and H. Kiiveri. Gaussian Markov distributions over finite graphs. *Ann. Statist.*, 14:138–150, 1986.

M. Sugiyama. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *J. Mach. Learn. Res.*, 8:1027–1061, 2007.

A. Szlam, M. Maggioni, and R. R. Coifman. Regularization on graphs with function-adapted diffusion process. *J. Mach. Learn. Res.*, 9:1711–1739, 2008.

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 945–952, 2001.

J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

S.A. Tomlins, R. Mehra, D.R. Rhodes, X. Cao, L. Wang, S.M. Dhanasekaran, S. Kalyana-Sundaram, J.T. Wei, M.A. Rubin, K.J. Pienta, R.B. Shah, and A.M. Chinnaiyan. Integrative molecular concept modeling of prostate cancer progression. *Nature Genetics*, 39(1):41–51, 2007.

G. Wahba. *Splines Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

Q. Wu, F. Liang, and S. Mukherjee. Regularized sliced inverse regression for kernel models. Technical Report 07-25, ISDS, Duke Univ., 2007.

Y. Xia, H. Tong, W. Li, and L-X. Zhu. An adaptive estimation of dimension reduction space. *J. Roy.Statist. Soc. Ser. B*, 64(3):363–410, 2002.

L. Zwald and G. Blanchard. On the convergence of eigenspaces in kernel principal component analysi s. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1649–1656. MIT Press, Cambridge, MA, 2006.

Journal of Machine Learning Research 11 (2010) 2199-2228 Submitted 11/09; Revised 5/10; Published 8/10

Regularized Discriminant Analysis, Ridge Regression and Beyond

Zhihua Zhang ZHZHANG@ZJU.EDU.CN
Guang Dai GUANG.GDAI@GMAIL.COM
Congfu Xu XUCONGFU@ZJU.EDU.CN
College of Computer Science & Technology
Zhejiang University
Hangzhou, Zhejiang 310027, China

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720-1776, USA

Editor: Inderjit Dhillon

Abstract

Fisher linear discriminant analysis (FDA) and its kernel extension—kernel discriminant analysis (KDA)—are well known methods that consider dimensionality reduction and classification jointly. While widely deployed in practical problems, there are still unresolved issues surrounding their efficient implementation and their relationship with least mean squares procedures. In this paper we address these issues within the framework of regularized estimation. Our approach leads to a flexible and efficient implementation of FDA as well as KDA. We also uncover a general relationship between regularized discriminant analysis and ridge regression. This relationship yields variations on conventional FDA based on the pseudoinverse and a direct equivalence to an ordinary least squares estimator.

Keywords: Fisher discriminant analysis, reproducing kernel, generalized eigenproblems, ridge regression, singular value decomposition, eigenvalue decomposition

1. Introduction

In this paper we are concerned with Fisher linear discriminant analysis (FDA), an enduring classification method in multivariate analysis and machine learning. It is well known that the FDA formulation reduces to the solution of a generalized eigenproblem (Golub and Van Loan, 1996) that involves the between-class scatter matrix and total scatter matrix of the data vectors. To solve the generalized eigenproblem, FDA typically requires the pooled scatter matrix to be nonsingular. This can become problematic when the dimensionality is high, because the scatter matrix is likely to be singular. In applications such as information retrieval, face recognition and microarray analysis, for example, we often meet undersampled problems which are in a "small n but large p" regime; that is, there are a small number of samples but a very large number of variables. There are two main variants of FDA in the literature that aim to deal with this issue: the *pseudoinverse* method and the *regularization* method (Hastie et al., 2001; Webb, 2002).

Another important family of methods for dealing with singularity is based on a two-stage process in which two symmetric eigenproblems are solved successively. This approach was pioneered by Kittler and Young (1973). Recently, Howland et al. (2003) used this approach to introduce the

generalized singular value decomposition (GSVD) (Paige and Saunders, 1981) into the FDA solution by using special representations of the pooled scatter matrix and between-class scatter matrix. A similar general approach has been used in the development of efficient approximation algorithms for FDA (Cheng et al., 1992; Ye et al., 2004). However, the challenge of developing an efficient general implementation methodology for FDA still remains.

In the binary classification problem, FDA is equivalent to a least mean squared error procedure (Duda et al., 2001). It is of great interest to obtain a similar relationship in multi-class problems. A significant literature has emerged to address this issue (Hastie et al., 2001; Park and Park, 2005b; Ye, 2007). However, the results obtained by these authors are subject to restrictive conditions. The problem of finding a general theoretical link between FDA and least mean squares is still open.

In this paper we address the issues within a regularization framework. We propose a novel algorithm for solving the regularized FDA (RFDA) problem. Our algorithm is more efficient than the GSVD-based algorithm (Howland et al., 2003), especially in the setting of "small n but large p" problems. More importantly, our algorithm leads us to an equivalence between RFDA and a ridge estimator for multivariate linear regression (Hoerl and Kennard, 1970). This equivalence is derived in a general setting and it is fully consistent with the established result in the binary problem (Duda et al., 2001).

Our algorithm is also appropriate for the pseudoinverse variant of FDA. Indeed, we establish an equivalence between the pseudoinverse variant and an ordinary least squares (OLS) estimation problem. Thus, we are able to resolve the open problem concerning the relationship between the multi-class FDA and multivariate linear estimation problems.

FDA relies on the assumption of linearity of the data manifold. In recent years, kernel methods (Shawe-Taylor and Cristianini, 2004) have aimed at removing such linearity assumptions. The kernel technology can circumvent the linearity assumption of FDA, because it works by nonlinearly mapping vectors in the input space to a higher-dimensional feature space and then implementing traditional versions of FDA in the feature space. Many different approaches have been proposed to extend FDA to kernel spaces in the existing literature (Baudat and Anouar, 2000; Mika et al., 2000; Roth and Steinhage, 2000).

The KDA method in Mika et al. (2000) was developed for binary problems only, and it was based on using the relationship between KDA and the least mean squared error procedure. A more general method, known as generalized discriminant analysis (GDA) (Baudat and Anouar, 2000), requires that the kernel matrix be nonsingular. Unfortunately, centering in the feature space will violate this requirement. Park and Park (2005a) argued that this might break down the theoretical justification for GDA and proposed their GSVD method to avoid this requirement for nonsingularity.

KDA methods have been successfully deployed in many practical problems. The approach to FDA that we present in the current paper not only handles the nonsingularity issue but also extends naturally to KDA, both in its regularization and pseudoinverse forms. We will see that our regularized KDA is different from the existing regularization methods for KDA (see, e.g., Park and Park, 2005a), where as we discuss later, there is a problem with inconsistency of solutions. Our methods for regularized KDA derive directly from the corresponding methods for regularized FDA and avoid the inconsistency problem.

Finally, we extend our approach for FDA as well as KDA to a certain family of generalized eigenvalue problems.

The paper is organized as follows. Section 2 reviews FDA and KDA, and Section 3 presents our KDA formulations. In Sections 4 and 5 we propose two new algorithms for FDA and KDA, respectively. An equivalence between FDA and multivariate linear regression problems is presented in Section 6. We conduct empirical comparisons in Section 7. We extend the approach to a certain family of generalized eigenproblems in Section 8 and conclude in Section 9.

2. Problem Formulation

We are concerned with a multi-class classification problem. Given a set of n p-dimensional data points, $\{\mathbf{x}_1,\ldots,\mathbf{x}_n\} \in \mathcal{X} \subset \mathbb{R}^p$, we assume that the $\mathbf{x}_i$ are to be grouped into c disjoint classes and that each $\mathbf{x}_i$ belongs to one and only one class. Let $V = \{1,2,\ldots,n\}$ denote the index set of the data points $\mathbf{x}_i$ and partition V into c disjoint subsets V_j; that is, $V_i \cap V_j = \varnothing$ for $i \neq j$ and $\cup_{j=1}^{c} V_j = V$, where the cardinality of V_j is n_j so that $\sum_{j=1}^{c} n_j = n$. We also make use of a matrix representation for the partitions. In particular, we let $\mathbf{E} = [e_{ij}]$ be an $n \times c$ indicator matrix with $e_{ij} = 1$ if input $\mathbf{x}_i$ is in class j and $e_{ij} = 0$ otherwise.

In this section we review FDA and KDA solutions to this multi-class classification problem. We begin by presenting our notation.

2.1 Preliminaries

Throughout this paper, $\mathbf{I}_m$ denotes the $m \times m$ identity matrix, $\mathbf{1}_m$ the $m \times 1$ vector of ones, $\mathbf{0}$ the zero vector or matrix with appropriate size, and $\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n'$ the $n \times n$ centering matrix. For an $m \times 1$ vector $\mathbf{a} = (a_1,\ldots,a_m)'$, $\mathrm{diag}(\mathbf{a})$ represents the $m \times m$ diagonal matrix with $a_1,\ldots,a_m$ as its diagonal entries. For an $m \times m$ matrix $\mathbf{A} = [a_{ij}]$, we let $\mathbf{A}^+$ be the Moore-Penrose inverse of $\mathbf{A}$, $\mathrm{tr}(\mathbf{A})$ be the trace of $\mathbf{A}$, $\mathrm{rk}(\mathbf{A})$ be the rank of $\mathbf{A}$ and $\|\mathbf{A}\|_F = \sqrt{\mathrm{tr}(\mathbf{A}'\mathbf{A})}$ be the Frobenius norm of $\mathbf{A}$. For an $m \times q$ real matrix $\mathbf{A}$, $\mathcal{R}(\mathbf{A})$ and $\mathcal{N}(\mathbf{A})$ denote its range and null spaces; that is, $\mathcal{R}(\mathbf{A}) = \{\mathbf{A}\mathbf{x} | \mathbf{x} \in \mathbb{R}^q\}$ and $\mathcal{N}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^q | \mathbf{A}\mathbf{x} = \mathbf{0}\}$.

For a matrix $\mathbf{A} \in \mathbb{R}^{m \times q}$ with $m \geq q$, we always express the (reduced) singular value decomposition (SVD) of $\mathbf{A}$ as $\mathbf{A} = \mathbf{U}\Gamma\mathbf{V}'$ where $\mathbf{U} \in \mathbb{R}^{m \times q}$ is a matrix with orthonormal columns (that is, $\mathbf{U}'\mathbf{U} = \mathbf{I}_q$), $\mathbf{V} \in \mathbb{R}^{q \times q}$ is orthogonal (i.e., $\mathbf{V}'\mathbf{V} = \mathbf{V}\mathbf{V}' = \mathbf{I}_q$), and $\Gamma = \mathrm{diag}(\gamma_1,\ldots,\gamma_q)$ is arrayed in descending order of $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_q$ (≥ 0). Let the rank of $\mathbf{A}$ be r ($\leq \min\{m,q\}$) and denote $\mathrm{rk}(\mathbf{A}) = r$. The condensed SVD of $\mathbf{A}$ is then $\mathbf{A} = \mathbf{U}_A\Gamma_A\mathbf{V}_A'$ where $\mathbf{U}_A \in \mathbb{R}^{m \times r}$ and $\mathbf{V}_A \in \mathbb{R}^{q \times r}$ are matrices with orthonormal columns (i.e., $\mathbf{U}_A'\mathbf{U}_A = \mathbf{I}_r$ and $\mathbf{V}_A'\mathbf{V}_A = \mathbf{I}_r$), and $\Gamma_A = \mathrm{diag}(\gamma_1,\ldots,\gamma_r)$ satisfies $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_r > 0$.

Given two matrices Σ_1 and $\Sigma_2 \in \mathbb{R}^{m \times m}$, we refer to $(\Lambda, \mathbf{B})$ where $\Lambda = \mathrm{diag}(\lambda_1,\ldots,\lambda_q)$ and $\mathbf{B} = [\mathbf{b}_1,\ldots,\mathbf{b}_q]$ as q eigenpairs of the matrix pencil (Σ_1,Σ_2) if $\Sigma_1\mathbf{B} = \Sigma_2\mathbf{B}\Lambda$; namely,

$$\Sigma_1\mathbf{b}_i = \lambda_i\Sigma_2\mathbf{b}_i, \quad \text{for } i = 1,\ldots,q.$$

The problem of finding eigenpairs of (Σ_1,Σ_2) is known as a *generalized eigenproblem*. In this paper, we especially consider the problem with the nonzero λ_i for $i = 1,\ldots,q$ and refer to $(\Lambda,\mathbf{B})$ as the nonzero eigenpairs of (Σ_1,Σ_2). If Σ_2 is nonsingular, $(\Lambda,\mathbf{B})$ is also referred to as the (nonzero) eigenpairs of $\Sigma_2^{-1}\Sigma_1$ because the generalized eigenproblem is equivalent to the eigenproblem:

$$\Sigma_2^{-1}\Sigma_1\mathbf{B} = \mathbf{B}\Lambda.$$

In the case that Σ_2 is singular, one typically resorts to a pseudoinverse eigenproblem:

$$\Sigma_2^{+}\Sigma_1\mathbf{B} = \mathbf{B}\Lambda.$$

Fortunately, we are able to establish a connection between the solutions of the generalized eigenproblem and its corresponding pseudoinverse eigenproblem. In particular, we have the following theorem, the proof of which is given in Appendix A.

Theorem 1 *Let Σ_1 and Σ_2 be two $m\times m$ real matrices. Assume $\mathcal{R}(\Sigma_1) \subseteq \mathcal{R}(\Sigma_2)$. Then, if $(\Lambda, \mathbf{B})$ are the nonzero eigenpairs of $\Sigma_2^+\Sigma_1$, we have that $(\Lambda, \mathbf{B})$ are the nonzero eigenpairs of the matrix pencil (Σ_1, Σ_2). Conversely, if $(\Lambda, \mathbf{B})$ are the nonzero eigenpairs of the matrix pencil (Σ_1, Σ_2), then $(\Lambda, \Sigma_2^+\Sigma_2\mathbf{B})$ are the nonzero eigenpairs of $\Sigma_2^+\Sigma_1$.*

As we see from Appendix A, a necessary and sufficient condition for $\mathcal{R}(\Sigma_1) \subseteq \mathcal{R}(\Sigma_2)$ is

$$\Sigma_2\Sigma_2^+\Sigma_1 = \Sigma_1.$$

Since $\mathbb{R}^m$ is equal to the direct sum of $\mathcal{R}(\Sigma_1)$ (or $\mathcal{R}(\Sigma_2)$) and $\mathcal{N}(\Sigma_1')$ (or $\mathcal{N}(\Sigma_2')$), we obtain that $\mathcal{R}(\Sigma_1) \subseteq \mathcal{R}(\Sigma_2)$ if and only if $\mathcal{N}(\Sigma_2') \subseteq \mathcal{N}(\Sigma_1')$. Furthermore, if both Σ_1 and Σ_2 are symmetric, then $\mathcal{N}(\Sigma_2) \subseteq \mathcal{N}(\Sigma_1)$ is equivalent to $\mathcal{R}(\Sigma_1) \subseteq \mathcal{R}(\Sigma_2)$.

2.2 Fisher Linear Discriminant Analysis

Let $\mathbf{m} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$ be the sample mean, and let $\mathbf{m}_j = \frac{1}{n_j}\sum_{i\in V_j} \mathbf{x}_i$ be the jth class mean for $j = 1,\ldots,c$. We then have the pooled scatter matrix $\mathbf{S}_t = \sum_{i=1}^n (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})'$ and the between-class scatter matrix $\mathbf{S}_b = \sum_{j=1}^c n_j(\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})'$. Conventional FDA solves the following generalized eigenproblem:

$$\mathbf{S}_b\mathbf{a}_j = \lambda_j\mathbf{S}_t\mathbf{a}_j, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_q > \lambda_{q+1} = 0 \tag{1}$$

where $q \leq \min\{p,\ c-1\}$ and where we refer to $\mathbf{a}_j$ as the jth discriminant direction. Note that we ignore a multiplier $1/n$ in these scatter matrices for simplicity.

Since $\mathbf{S}_t = \mathbf{S}_b + \mathbf{S}_w$ where $\mathbf{S}_w$ is the pooled within-class scatter matrix, FDA is equivalent to finding a solution to

$$\mathbf{S}_b\mathbf{a} = \lambda/(1-\lambda)\mathbf{S}_w\mathbf{a}.$$

We see that FDA involves solving the generalized eigenproblem in (1), which can be expressed in matrix form:

$$\mathbf{S}_b\mathbf{A} = \mathbf{S}_t\mathbf{A}\Lambda. \tag{2}$$

Here $\mathbf{A} = [\mathbf{a}_1,\ldots,\mathbf{a}_q]$ $(n\times q)$ and $\Lambda = \mathrm{diag}(\lambda_1,\ldots,\lambda_q)$ $(q\times q)$. If $\mathbf{S}_t$ is nonsingular, we obtain

$$\mathbf{S}_t^{-1}\mathbf{S}_b\mathbf{A} = \mathbf{A}\Lambda.$$

Thus, the $(\lambda_j, \mathbf{a}_j)$ are the eigenpairs of $\mathbf{S}_t^{-1}\mathbf{S}_b$ and the eigenvectors corresponding to the largest eigenvalues of $\mathbf{S}_t^{-1}\mathbf{S}_b$ are used for the discriminant directions. Since $\mathrm{rk}(\mathbf{S}_b)$ is at most $c-1$, the projection will be onto a space of dimension at most $c-1$ (i.e., $q \leq c-1$).

In applications such as information retrieval, face recognition and microarray analysis, however, we often meet a "small n but large p" problem. Thus, $\mathbf{S}_t$ is usually ill-conditioned; that is, it is either singular or close to singular. In this case, $\mathbf{S}_t^{-1}\mathbf{S}_b$ is not well defined or cannot be computed accurately.

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]'$ $(n \times p)$, $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_c]'$ $(c \times p)$, $\Pi = \mathrm{diag}(n_1, \ldots, n_c)$ $(c \times c)$, $\Pi^{\frac{1}{2}} = \mathrm{diag}(\sqrt{n}_1, \ldots, \sqrt{n}_c)$, $\pi = (n_1, \ldots, n_c)'$, $\sqrt{\pi} = (\sqrt{n}_1, \ldots, \sqrt{n}_c)'$ and $\mathbf{H}_\pi = \mathbf{I}_c - \frac{1}{n}\sqrt{\pi}\sqrt{\pi}'$. It then follows that $\mathbf{1}_n'\mathbf{E} = \mathbf{1}_c'\Pi = \pi'$, $\mathbf{E}\mathbf{1}_c = \mathbf{1}_n$, $\mathbf{1}_c'\pi = n$, $\mathbf{E}'\mathbf{E} = \Pi$, $\Pi^{-1}\pi = \mathbf{1}_c$, and

$$\mathbf{M} = \Pi^{-1}\mathbf{E}'\mathbf{X}.$$

In addition, we have

$$\mathbf{E}\Pi^{-\frac{1}{2}}\mathbf{H}_\pi = \mathbf{H}\mathbf{E}\Pi^{-\frac{1}{2}}$$

given that $\mathbf{E}\Pi^{-\frac{1}{2}}\mathbf{H}_\pi = \mathbf{E}\Pi^{-\frac{1}{2}} - \frac{1}{n}\mathbf{1}_n\sqrt{\pi}'$ and $\mathbf{H}\mathbf{E}\Pi^{-\frac{1}{2}} = \mathbf{E}\Pi^{-\frac{1}{2}} - \frac{1}{n}\mathbf{1}_n\sqrt{\pi}'$.

Based on these results and the idempotency of $\mathbf{H}$, $\mathbf{S}_t$ can be written as

$$\mathbf{S}_t = \mathbf{X}'\mathbf{H}\mathbf{H}\mathbf{X} = \mathbf{X}'\mathbf{H}\mathbf{X}, \tag{3}$$

and we have

$$\begin{aligned}
\mathbf{S}_b &= \mathbf{M}'\left[\Pi - \frac{1}{n}\pi\pi'\right]\mathbf{M} \\
&= \mathbf{M}'\left[\Pi^{\frac{1}{2}} - \frac{1}{n}\pi\sqrt{\pi}'\right]\left[\Pi^{\frac{1}{2}} - \frac{1}{n}\sqrt{\pi}\pi'\right]\mathbf{M} \\
&= \mathbf{X}'\mathbf{E}\Pi^{-1}\Pi^{\frac{1}{2}}\mathbf{H}_\pi\mathbf{H}_\pi\Pi^{\frac{1}{2}}\Pi^{-1}\mathbf{E}'\mathbf{X} \\
&= \mathbf{X}'\mathbf{H}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{H}\mathbf{X}.
\end{aligned} \tag{4}$$

Given these representations of $\mathbf{S}_t$ and $\mathbf{S}_b$, the problem in (2) can be solved by using the GSVD method (Van Loan, 1976; Paige and Saunders, 1981; Golub and Van Loan, 1996; Howland et al., 2003).

There are also two variants of conventional FDA in the literature that aim to handle the ill-conditioned problem (Webb, 2002). The first variant, the *pseudoinverse method*, involves replacing $\mathbf{S}_t^{-1}$ by $\mathbf{S}_t^+$ and solving the following eigenproblem:

$$\mathbf{S}_t^+\mathbf{S}_b\mathbf{A} = \mathbf{A}\Lambda. \tag{5}$$

Note that $\mathbf{S}_t^+$ exists and is unique (Golub and Van Loan, 1996). Moreover, $\mathbf{S}_t^+$ is equal to $\mathbf{S}_t^{-1}$ whenever $\mathbf{S}_t$ is nonsingular. Thus, we will use (5) when $\mathbf{S}_t$ is either nonsingular or singular.

The second variant is referred to as *regularized discriminant analysis* (RDA) (Friedman, 1989). It replaces $\mathbf{S}_t$ by $\mathbf{S}_t + \sigma^2\mathbf{I}_p$ and solves the following eigenproblem:

$$(\mathbf{S}_t + \sigma^2\mathbf{I}_p)^{-1}\mathbf{S}_b\mathbf{A} = \mathbf{A}\Lambda. \tag{6}$$

It is a well known result that FDA is equivalent to a least mean squared error procedure in the binary classification problem ($c = 2$) (Duda et al., 2001). Recently, similar relationships have been studied for multi-class ($c > 2$) problems (Hastie et al., 2001; Park and Park, 2005b; Ye, 2007). Moreover, Park and Park (2005b) proposed an efficient algorithm for FDA based on a least mean squared error procedure in the multi-class problem.

We can see that the solution $\mathbf{A}$ for (5) or (6) is not unique. For example, if $\mathbf{A}$ is the solution, then so is $\mathbf{A}\mathbf{D}$ where $\mathbf{D}$ is an arbitrary $q \times q$ nonsingular diagonal matrix. Thus, the constraint $\mathbf{A}'(\mathbf{S}_t + \sigma^2\mathbf{I}_p)\mathbf{A} = \mathbf{I}_q$ is typically imposed in the literature. In this paper we concentrate on the solution of (6) with or without this constraint, and investigate the connection with a ridge regression problem in the multi-class setting.

2.3 Kernel Discriminant Analysis

Kernel methods (Shawe-Taylor and Cristianini, 2004) work in a feature space $\mathcal{F}$, which is related to the original input space $\mathcal{X} \subset \mathbb{R}^p$ by a mapping,

$$\varphi : \mathcal{X} \to \mathcal{F}.$$

That is, φ is a vector-valued function which gives a vector $\varphi(\mathbf{s})$, called a *feature vector*, corresponding to an input $\mathbf{s} \in \mathcal{X}$. In kernel methods, we are given a reproducing kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $K(\mathbf{s},\mathbf{t}) = \varphi(\mathbf{s})'\varphi(\mathbf{t})$ for $\mathbf{s},\mathbf{t} \in \mathcal{X}$. The mapping $\varphi(\cdot)$ itself is typically not given explicitly.

In the sequel, we use the tilde notation to denote vectors and matrices in the feature space. For example, the data vectors and mean vectors in the feature space are denoted as $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{m}}_j$. Accordingly, $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1,\ldots,\tilde{\mathbf{x}}_n]'$ $(n\times g)$ and $\tilde{\mathbf{M}} = [\tilde{\mathbf{m}}_1,\ldots,\tilde{\mathbf{m}}_c]'$ $(c\times g)$ are the data and mean matrices in the feature space. Here g is the dimension of the feature space. Although g is possibly infinite, we here assume that it is finite but not necessarily known.

Kernel discriminant analysis (KDA) seeks to solve the following generalized eigenproblem:

$$\tilde{\mathbf{S}}_b\tilde{\mathbf{A}} = \tilde{\mathbf{S}}_t\tilde{\mathbf{A}}\Lambda, \tag{7}$$

where $\tilde{\mathbf{S}}_t$ and $\tilde{\mathbf{S}}_b$ are the pooled scatter matrix and the between-class scatter matrix in $\mathcal{F}$, respectively:

$$\begin{aligned}
\tilde{\mathbf{S}}_t &= \sum_{i=1}^{n}(\tilde{\mathbf{x}}_i-\tilde{\mathbf{m}})(\tilde{\mathbf{x}}_i-\tilde{\mathbf{m}})' = \tilde{\mathbf{X}}'\mathbf{H}\tilde{\mathbf{X}},\\
\tilde{\mathbf{S}}_b &= \sum_{j=1}^{c} n_j(\tilde{\mathbf{m}}_j-\tilde{\mathbf{m}})(\tilde{\mathbf{m}}_j-\tilde{\mathbf{m}})' = \tilde{\mathbf{X}}'\mathbf{H}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{H}\tilde{\mathbf{X}}.
\end{aligned}$$

The KDA problem is to solve (7), doing so by working solely with the kernel matrix $\mathbf{K} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}'$. This is done by noting that $\tilde{\mathbf{A}}$ can be expressed as

$$\tilde{\mathbf{A}} = \sum_{i=1}^{n}(\tilde{\mathbf{x}}_i-\tilde{\mathbf{m}})\beta_i' + \mathbf{N} = \tilde{\mathbf{X}}'\mathbf{H}\Upsilon + \mathbf{N}, \tag{8}$$

where $\Upsilon = [\beta_1,\ldots,\beta_n]$ $(n\times q)$ and $\mathbf{N} \in \mathbb{R}^{g\times q}$ such that $\mathbf{N}'\tilde{\mathbf{X}}'\mathbf{H} = \mathbf{0}$ (Park and Park, 2005a; Mika et al., 2000). It then follows from (7) that

$$\tilde{\mathbf{X}}'\mathbf{H}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H}\Upsilon = \tilde{\mathbf{X}}'\mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H}\Upsilon\Lambda. \tag{9}$$

This implies that $(\Lambda, \tilde{\mathbf{X}}'\mathbf{H}\Upsilon)$ are also the q eigenpairs of the matrix pencil $(\tilde{\mathbf{S}}_b, \tilde{\mathbf{S}}_t)$. Thus, in the literature the solution of (7) is typically restricted to $\mathcal{R}(\tilde{\mathbf{X}}'\mathbf{H})$; that is, $\mathbf{N} = \mathbf{0}$ is set.

Premultiplying both sides of the Equation (9) by $\mathbf{H}\tilde{\mathbf{X}}$, we have a new generalized eigenvalue problem

$$\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \mathbf{C}\mathbf{C}\Upsilon\Lambda, \tag{10}$$

which involves only the kernel matrix $\mathbf{K} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}'$ via its centered form $\mathbf{C} = \mathbf{H}\mathbf{K}\mathbf{H}$.

The current concern then becomes that of solving the problem (10). Although $\mathbf{K}$ can be assumed to be nonsingular, $\mathbf{C}$ is positive semidefinite but not positive definite because the centering matrix

$\mathbf{H}$ is singular. In fact, the rank of $\mathbf{C}$ is not larger than $n-1$ because the rank of $\mathbf{H}$ is $n-1$. Thus the GDA method devised by Baudat and Anouar (2000) cannot be used directly for problem (10).

To address this problem Park and Park (2005a) proposed a GSVD-based algorithm to solve (10). Running this algorithm requires the complete orthogonal decomposition (Golub and Van Loan, 1996) of matrix $[\mathbf{CE\Pi}^{-\frac{1}{2}}, \mathbf{C}]'$, which is of size $(n+c)\times n$. This approach is infeasible for large values of n. Thus, Park and Park (2005a) developed an efficient alternative which consists of two SVD procedures but does not involve the complete orthogonal decomposition of an $(n+c)\times n$ matrix. We refer to it as the *SVD-based algorithm*.

Another approach to solving the problem (10) is based on the following regularized version of the problem:

$$\mathbf{CE\Pi}^{-1}\mathbf{E}'\mathbf{C}\Upsilon = (\mathbf{CC} + \sigma^2\mathbf{I}_n)\Upsilon\Lambda, \tag{11}$$

which was also studied by Park and Park (2005a). Note that this variant is not a directly regularized form of the original KDA problem in (7).

After having obtained Υ from (10) or (11), for a new input vector $\mathbf{x}$, the projection $\mathbf{z}$ of its feature vector $\tilde{\mathbf{x}}$ onto $\tilde{\mathbf{A}}$ is computed by

$$\mathbf{z} = \Upsilon'\mathbf{H}\tilde{\mathbf{X}}\Big(\tilde{\mathbf{x}} - \frac{1}{n}\tilde{\mathbf{X}}'\mathbf{1}_n\Big) = \Upsilon'\mathbf{H}\Big(\mathbf{k}_\mathbf{x} - \frac{1}{n}\mathbf{K}\mathbf{1}_n\Big), \tag{12}$$

where $\mathbf{k}_\mathbf{x} = \big(K(\mathbf{x},\mathbf{x}_1),\ldots,K(\mathbf{x},\mathbf{x}_n)\big)'$. This shows that the kernel trick can be used for KDA, and this approach has been widely deployed in practical problems. However, a theoretical justification for using the projection in (12) has been lacking in the literature. We are able to provide such as justification as follows. Recall that if $(\Lambda, \tilde{\mathbf{X}}'\mathbf{H}\Upsilon)$ is the solution of (7), then (Λ,Υ) is the solution of (10). As we will see in Theorem 3, if (Λ,Υ) is the solution of (10), $(\Lambda, \tilde{\mathbf{X}}'\mathbf{H}\Upsilon)$ is the solution of (7). This justifies the projection (12).

Note, however, that if (Λ,Υ) is the solution of (11) this does not imply that $(\Lambda, \tilde{\mathbf{X}}'\mathbf{H}\Upsilon)$ is the solution of (7). This shows that the projection (12) is inconsistent with (11). This is an inconsistency that underlies the regularized KDA methodology of Park and Park (2005a). The new methodology that we propose in the following section surmounts this problem.

3. New Approaches to Kernel Discriminant Analysis

Recall that we are interested in the pseudoinverse and regularization forms of (7), defined respectively by

$$\tilde{\mathbf{S}}_t^+\tilde{\mathbf{S}}_b\tilde{\mathbf{A}} = \tilde{\mathbf{A}}\Lambda \tag{13}$$

and

$$\tilde{\mathbf{S}}_b\tilde{\mathbf{A}} = (\tilde{\mathbf{S}}_t + \sigma^2\mathbf{I}_g)\tilde{\mathbf{A}}\Lambda. \tag{14}$$

We wish to find solutions of (7) and (13) or (14) that are consistent with each other.

Since $\mathcal{R}(\tilde{\mathbf{S}}_t) = \mathcal{R}(\tilde{\mathbf{X}}'\mathbf{H}\mathbf{H}\tilde{\mathbf{X}}) = \mathcal{R}(\tilde{\mathbf{X}}'\mathbf{H})$ and $\mathcal{R}(\tilde{\mathbf{S}}_b) = \mathcal{R}(\tilde{\mathbf{X}}'\mathbf{H}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{H}\tilde{\mathbf{X}}) = \mathcal{R}(\tilde{\mathbf{X}}'\mathbf{H}\mathbf{E})$, we have that $\mathcal{R}(\tilde{\mathbf{S}}_b) \subseteq \mathcal{R}(\tilde{\mathbf{S}}_t)$. As a direct corollary of Theorem 1, we thus obtain a connection between (7) and (13); that is,

Theorem 2 *If $(\Lambda,\tilde{\mathbf{A}})$ (nonzero eigenpairs) is the solution of (13), then $(\Lambda,\tilde{\mathbf{A}})$ is the solution of (7). Conversely, if $(\Lambda,\tilde{\mathbf{A}})$ is the solution of (7), then $(\Lambda,\tilde{\mathbf{S}}_t^+\tilde{\mathbf{S}}_t\tilde{\mathbf{A}})$ is the solution of (13).*

Theorem 2 implies that the solution of (7) can be obtained from (13). To solve (13), we consider the pseudoinverse form of (10), which is

$$\mathbf{C}^+\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \Upsilon\Lambda \tag{15}$$

due to $\mathbf{C}^+\mathbf{C}^+\mathbf{C} = \mathbf{C}^+$ (see Lemma 11). To obtain the solution of (14), we substitute (8) into (14) and then premultiply by $\mathbf{H}\tilde{\mathbf{X}}$. As a result, we have

$$\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = (\mathbf{C}\mathbf{C} + \sigma^2\mathbf{C})\Upsilon\Lambda. \tag{16}$$

The following theorem shows that we are able to obtain the solutions of (7), (13) and (14) respectively from the solutions of (10), (15) and (16). That is,

Theorem 3 *Considering the KDA problems, we have:*

(i) *If* (Λ,Υ) *is the solution of (10), then* $\left(\Lambda,\tilde{\mathbf{X}}'\mathbf{H}\Upsilon\right)$ *is the solution of (7).*

(ii) *If* (Λ,Υ) *is the solution of (15), then* $\left(\Lambda,\tilde{\mathbf{X}}'\mathbf{H}\Upsilon\right)$ *is the solution of (13).*

(iii) *If* (Λ,Υ) *is the solution of (16), then* $\left(\Lambda,\tilde{\mathbf{X}}'\mathbf{H}\Upsilon\right)$ *is the solution of (14).*

The proof of this theorem is given in Appendix C. Theorem 3 shows that the solutions of (7), (13) and (14) lie in span$\{\tilde{\mathbf{X}}'\mathbf{H}\}$. Moreover, we see that $\left(\Lambda,\tilde{\mathbf{X}}'\mathbf{H}\Upsilon\right)$ are their solutions. We also note that (16) is different from (11). Theorem 3 provides a relationship between (14) and (16); there is not a similar relationship between (14) and (11).

Finally, as a corollary of Theorems 2 and 3, we have

Corollary 4 *If* (Λ,Υ) *is the solution of (15), then* $\left(\Lambda,\tilde{\mathbf{X}}'\mathbf{H}\Upsilon\right)$ *is the solution of (7). Moreover, if* (Λ,Υ) *is the solution of (10), then* $\left(\Lambda,\tilde{\mathbf{X}}'\mathbf{H}\Upsilon\right)$ *is the solution of (13).*

We see from Corollary 4 that the solution of (7) can be also obtained from (15). We now concentrate our attention on the regularized KDA problem (14). We first handle (14) by using (16) and Theorem 3, and then we present an approach for directly solving (14).

4. SVD-based Algorithms for RDA Problems

It is clear that we can solve the regularized KDA (RKDA) problem in (11) by solving

$$(\mathbf{C}\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \Upsilon\Lambda.$$

However, since $\mathbf{C}\mathbf{C} + \sigma^2\mathbf{C}$ is singular, this approach is not appropriate for the RKDA problem in (16). In this section we show how to solve the RKDA problems in both (11) and (16), as well as the regularized FDA (RFDA) problem in (6).

In particular, we exploit the SVD-based algorithm developed for solving (10) by Park and Park (2005a). Our algorithms are summarized as Algorithm 1, which is an SVD-based method for solving RFDA problem (6), and Algorithms 2 and 3, which are SVD-based algorithms for RKDA problems (11) and (16). We defer the derivations to Section 8 in which we consider the SVD-based algorithm for a more general generalized eigenvalue problem. According to Theorem 3 and the following theorem, we immediately have the solution of (14) as $\tilde{\mathbf{A}} = \tilde{\mathbf{X}}'\mathbf{H}\Upsilon$.

Algorithm 1 SVD-based Algorithm for RFDA problem (6)

procedure RFDA($\mathbf{X}, \mathbf{E}, \pi, \Pi, \sigma^2$)
  Perform the condensed SVD of $\mathbf{HX}$ as $\mathbf{HX} = \mathbf{U}_X \Gamma_X \mathbf{V}_X'$;
  Calculate $\mathbf{F} = (\Gamma_X^2 + \sigma^2 \mathbf{I}_r)^{-\frac{1}{2}} \Gamma_X \mathbf{U}_X' \mathbf{E} \Pi^{-\frac{1}{2}}$ where $r = \mathrm{rk}(\Gamma_X)$;
  Perform the condensed SVD of $\mathbf{F}$ as $\mathbf{F} = \mathbf{U}_F \Gamma_F \mathbf{V}_F'$ and set $q = \mathrm{rk}(\Gamma_F)$;
  Return $\mathbf{A} = \mathbf{V}_X (\Gamma_X^2 + \sigma^2 \mathbf{I}_r)^{-\frac{1}{2}} \mathbf{U}_F$ as the solution of RFDA.
end procedure

Algorithm 2 SVD-based Algorithm for RKDA problem (11)

procedure RKDA($\mathbf{C}, \mathbf{E}, \pi, \Pi, \sigma^2$)
  Perform the condensed SVD of $\mathbf{C}$ as $\mathbf{C} = \mathbf{U}_C \Gamma_C \mathbf{U}_C'$;
  Calculate $\mathbf{F} = (\Gamma_C^2 + \sigma^2 \mathbf{I}_r)^{-\frac{1}{2}} \Gamma_C \mathbf{U}_C' \mathbf{E} \Pi^{-\frac{1}{2}}$ where $r = \mathrm{rk}(\Gamma_C)$;
  Perform the condensed SVD of $\mathbf{F}$ as $\mathbf{F} = \mathbf{U}_F \Gamma_F \mathbf{V}_F'$;
  Let $\Upsilon = \mathbf{U}_C (\Gamma_C^2 + \sigma^2 \mathbf{I}_r)^{-\frac{1}{2}} \mathbf{U}_F$ and set $q = \mathrm{rk}(\Gamma_F)$;
  Calculate $\mathbf{z}$ via (12) as the q-dimensional representation of $\mathbf{x}$.
end procedure

Theorem 5 *Consider Algorithms 1, 2 and 3 for the corresponding RDA problems.*

(i) *If* $\mathbf{A}$ *is obtained from Algorithm 1, then,*

$$\mathbf{A}'(\mathbf{S}_t + \sigma^2 \mathbf{I}_p)\mathbf{A} = \mathbf{I}_q \quad \textit{and} \quad \mathbf{A}'\mathbf{S}_b\mathbf{A} = \Gamma_F^2.$$

(ii) *If* Υ *is obtained from Algorithm 2, then*

$$\Upsilon'(\mathbf{CC} + \sigma^2 \mathbf{I}_n)\Upsilon = \mathbf{I}_q \quad \textit{and} \quad \Upsilon'\mathbf{CE}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \Gamma_F^2.$$

(iii) *If* Υ *is obtained from Algorithm 3 and* $\tilde{\mathbf{A}} = \tilde{\mathbf{X}}'\mathbf{H}\Upsilon$*, then*

$$\Upsilon'(\mathbf{CC} + \sigma^2 \mathbf{C})\Upsilon = \mathbf{I}_q \quad \textit{and} \quad \Upsilon'\mathbf{CE}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \Gamma_F^2,$$

and

$$\tilde{\mathbf{A}}'(\tilde{\mathbf{S}}_t + \sigma^2 \mathbf{I}_g)\tilde{\mathbf{A}} = \mathbf{I}_q \quad \textit{and} \quad \tilde{\mathbf{A}}'\tilde{\mathbf{S}}_b\tilde{\mathbf{A}} = \Gamma_F^2.$$

The proof of the theorem is given in Appendix D. According to this theorem, for a new $\mathbf{x}$, the projection $\mathbf{z}$ onto $\tilde{\mathbf{A}}$ is given by

$$\mathbf{z} = \Upsilon'\mathbf{H}\Big(\mathbf{k}_\mathbf{x} - \frac{1}{n}\mathbf{K}\mathbf{1}_n\Big). \tag{17}$$

Note that if $\sigma^2 = 0$, Algorithm 1 degenerates to the SVD-based algorithm for the conventional FDA in (2) (Howland et al., 2003) and Algorithms 2 and 3 become identical.

5. EVD-based Algorithms for RDA

It is desirable to directly find the solution of the RKDA problem (14), rather than obtaining the solution indirectly via (16). However, it is not feasible to devise a SVD-based algorithm for directly solving the RKDA problem in (14). In this section we propose a new approach to solving the RFDA problem (6). We then extend this approach for the solution of the RKDA problem (14).

Algorithm 3 SVD-based Algorithm for RKDA problem (16) as well as for (14)

procedure RKDA($\mathbf{C}, \mathbf{E}, \pi, \Pi, \sigma^2$)
  Perform the condensed SVD of $\mathbf{C}$ as $\mathbf{C} = \mathbf{U}_C \Gamma_C \mathbf{U}_C'$;
  Calculate $\mathbf{F} = (\Gamma_C^2 + \sigma^2 \Gamma_C)^{-\frac{1}{2}} \Gamma_C \mathbf{U}_C' \mathbf{E} \Pi^{-\frac{1}{2}}$ where $r = \mathrm{rk}(\Gamma_C)$;
  Perform the condensed SVD of $\mathbf{F}$ as $\mathbf{F} = \mathbf{U}_F \Gamma_F \mathbf{V}_F'$;
  Let $\Upsilon = \mathbf{U}_C (\Gamma_C^2 + \sigma^2 \Gamma_C)^{-\frac{1}{2}} \mathbf{U}_F$ and set $q = \mathrm{rk}(\Gamma_F)$;
  Calculate $\mathbf{z}$ via (17) as the q-dimensional representation of $\mathbf{x}$.
end procedure

5.1 The Algorithm for RFDA

We reformulate the eigenproblem in (6) as

$$\mathbf{G}\Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{H}\mathbf{X}\mathbf{A} = \mathbf{A}\Lambda,$$

where

$$\mathbf{G} = (\mathbf{X}'\mathbf{H}\mathbf{X} + \sigma^2 \mathbf{I}_p)^{-1} \mathbf{X}'\mathbf{H}\mathbf{E}\Pi^{-\frac{1}{2}} \tag{18}$$

due to (3) and (4). We also have

$$\mathbf{G} = \mathbf{X}'\mathbf{H}(\mathbf{H}\mathbf{X}\mathbf{X}'\mathbf{H} + \sigma^2 \mathbf{I}_n)^{-1} \mathbf{E}\Pi^{-\frac{1}{2}} \tag{19}$$

due to $(\mathbf{X}'\mathbf{H}\mathbf{X} + \sigma^2 \mathbf{I}_p)^{-1}\mathbf{X}'\mathbf{H} = \mathbf{X}'\mathbf{H}(\mathbf{H}\mathbf{X}\mathbf{X}'\mathbf{H} + \sigma^2 \mathbf{I}_n)^{-1}$. If $n < p$, we can use (19) to reduce the computational cost. Moreover, we will see that (19) plays a key role in the development of an efficient algorithm for KDA to be presented shortly.

Let $\mathbf{R} = \Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{H}\mathbf{X}\mathbf{G}$. Since $\mathbf{G}\Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{H}\mathbf{X}$ ($p \times p$) and $\mathbf{R}$ ($c \times c$) have the same nonzero eigenvalues (Horn and Johnson, 1985), the λ_j, $j = 1, \ldots, q$, are the nonzero eigenvalues of $\mathbf{R}$. Moreover, if $(\Lambda, \mathbf{V})$ is the nonzero eigenpair of $\mathbf{R}$, $(\Lambda, \mathbf{G}\mathbf{V})$ is the nonzero eigenpair of $\mathbf{G}\Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{H}\mathbf{X}$. Note that

$$\mathbf{R} = \Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{H}\mathbf{X}(\mathbf{X}'\mathbf{H}\mathbf{X} + \sigma^2 \mathbf{I}_p)^{-1}\mathbf{X}'\mathbf{H}\mathbf{E}\Pi^{-\frac{1}{2}}. \tag{20}$$

This shows that $\mathbf{R}$ is positive semidefinite.

We use these facts to develop an algorithm for solving the RFDA problem in (6). This is also a two-step process, which is presented in Algorithm 4. The first step computes $(\sigma^2 \mathbf{I}_p + \mathbf{X}'\mathbf{H}\mathbf{X})^{-1}$ (or $(\sigma^2 \mathbf{I}_n + \mathbf{H}\mathbf{X}\mathbf{X}'\mathbf{H})^{-1}$), while the second step is an SVD procedure. Note that the first step can be implemented by computing the condensed SVD of $\mathbf{X}'\mathbf{H}\mathbf{X}$ (or $\mathbf{H}\mathbf{X}\mathbf{X}'\mathbf{H}$). Since $\mathbf{R}$ and $\mathbf{X}'\mathbf{H}\mathbf{X}$ ($\mathbf{H}\mathbf{X}\mathbf{X}'\mathbf{H}$) are positive semidefinite, their SVD are equivalent to the eigenvalue decomposition (EVD). Thus, we refer to this two-step process as an *EVD-based algorithm*, distinguishing it from the SVD-based algorithm.

The first step calculates $\mathbf{G}$ by either (18) or (19). The computational complexity is $O(m^3)$ where $m = \min(n, p)$. The second step forms the condensed SVD of $\mathbf{R}$ for which the computational complexity is $O(c^3)$. If both n and p are large, we recommend an approximate strategy; that is, we first perform the incomplete Cholesky decomposition of $\mathbf{H}\mathbf{X}\mathbf{X}'\mathbf{H}$ (or $\mathbf{X}'\mathbf{H}\mathbf{X}$) and then calculate $(\mathbf{H}\mathbf{X}\mathbf{X}'\mathbf{H} + \sigma^2 \mathbf{I}_n)^{-1}$ (or $(\mathbf{X}'\mathbf{H}\mathbf{X} + \sigma^2 \mathbf{I}_p)^{-1}$) via the Sherman-Morrison-Woodbury formula (Golub and Van Loan, 1996). This strategy makes the first step still efficient.

When $\sigma^2 = 0$, we can solve the problem in (5) by simply adjusting the first step in the EVD-based algorithm. In particular, we calculate $\mathbf{G}$ by

$$\begin{aligned}\mathbf{G} &= (\mathbf{X}'\mathbf{H}\mathbf{X})^{+}\mathbf{X}'\mathbf{H}\mathbf{E}\Pi^{-\frac{1}{2}} \\ &\overset{(or)}{=} \mathbf{X}'\mathbf{H}(\mathbf{H}\mathbf{X}\mathbf{X}'\mathbf{H})^{+}\mathbf{E}\Pi^{-\frac{1}{2}}. \end{aligned} \tag{21}$$

Algorithm 4 EVD-based Algorithm for RFDA problem (6)

1: **procedure** RFDA($\mathbf{X}, \mathbf{E}, \Pi, \sigma^2$)
2: Calculate $\mathbf{G}$ by (18) or (19) and $\mathbf{R}$ by (20);
3: Perform the condensed SVD of $\mathbf{R}$ as $\mathbf{R} = \mathbf{V}_R\Gamma_R\mathbf{V}_R'$;
4: Return $\mathbf{A} = \mathbf{G}\mathbf{V}_R\Gamma_R^{-\frac{1}{2}}$ or $\mathbf{B} = \mathbf{G}\mathbf{V}_R$ as the solution of RFDA problem (6).
5: **end procedure**

Compared with the SVD-based algorithm, the EVD-based algorithm is more efficient, especially for "small n but large p" problems. Using the notation in Algorithms 1 and 4, we have

$$\mathbf{R} = \mathbf{F}'\mathbf{F}$$

by performing some matrix computations. This implies that $\Gamma_R = \Gamma_F^2$. Moreover, it is immediate to obtain the following result.

Theorem 6 *Let* $\mathbf{A}$ *be obtained from Algorithm 4. Then,*

$$\mathbf{A}'(\mathbf{S}_t + \sigma^2\mathbf{I}_p)\mathbf{A} = \mathbf{I}_q \quad \textit{and} \quad \mathbf{A}'\mathbf{S}_b\mathbf{A} = \Gamma_F^2.$$

This theorem shows that Algorithms 1 and 4 are essentially equivalent. As mentioned before, it is not feasible to develop an SVD-based algorithm for solving the RKDA problem (14), which is the kernel extension of RFDA in (6). On the other hand, in the next subsection we will see that Algorithm 4 can be used for solving the RKDA problem (14).

5.2 The Algorithm for RKDA

It follows immediately from (19) that

$$\tilde{\mathbf{G}} = \tilde{\mathbf{X}}'\mathbf{H}(\mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{E}\Pi^{-\frac{1}{2}}$$

from which, using (20), we calculate $\mathbf{R}$ by

$$\mathbf{R} = \Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{C}(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{E}\Pi^{-\frac{1}{2}}.$$

Moreover, given a new input vector $\mathbf{x}$, we can compute the projection $\mathbf{z}$ of the feature vector $\tilde{\mathbf{x}}$ onto $\tilde{\mathbf{A}}$ through

$$\begin{aligned}\mathbf{z} &= \tilde{\mathbf{A}}'(\tilde{\mathbf{x}} - \tilde{\mathbf{m}}) \\ &= \Gamma_R^{-\frac{1}{2}}\mathbf{V}_R'\Pi^{-\frac{1}{2}}\mathbf{E}'(\mathbf{C} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{H}\tilde{\mathbf{X}}\Big(\tilde{\mathbf{x}} - \frac{1}{n}\tilde{\mathbf{X}}'\mathbf{1}_n\Big) \\ &= \Gamma_R^{-\frac{1}{2}}\mathbf{V}_R'\Pi^{-\frac{1}{2}}\mathbf{E}'(\mathbf{C} + \sigma^2\mathbf{I}_n)^{-1}\mathbf{H}\Big(\mathbf{k}_{\mathbf{x}} - \frac{1}{n}\mathbf{K}\mathbf{1}_n\Big). \end{aligned} \tag{22}$$

This shows that we can calculate $\mathbf{R}$ and $\mathbf{z}$ directly using $\mathbf{K}$ and $\mathbf{k_x}$. We thus obtain an EVD-based algorithm for RKDA, which is given in Algorithm 5. Also, when $\sigma^2 = 0$, we calculate $\mathbf{R}$ by

$$\mathbf{R} = \Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{C}\mathbf{C}^+\mathbf{E}\Pi^{-\frac{1}{2}}$$

and exploit the EVD-based algorithm to solve the following variant of KDA:

$$\tilde{\mathbf{S}}_t^+\tilde{\mathbf{S}}_b\tilde{\mathbf{A}} = \tilde{\mathbf{A}}\Lambda.$$

We see that the EVD-based algorithm is more efficient than the SVD-based algorithm (i.e., Algorithm 2) for the RKDA problem in (11). Recall that the RKDA problem (14) is not fully equivalent to that in (11). Moreover, we also have an EVD-based algorithm for solving (11), by replacing $\mathbf{C}$ by $\mathbf{C}\mathbf{C}$ in calculating $\mathbf{R}$ and (22) by (17) in calculating $\mathbf{z}$. However, the resulting algorithm is less efficient computationally.

Algorithm 5 EVD-based Algorithm for RKDA problem (14)

1: **procedure** RKDA($\mathbf{K}, \mathbf{E}, \mathbf{k_x}, \Pi, \sigma^2$)
2: Calculate $\mathbf{R} = \Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{C}(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{E}\Pi^{-\frac{1}{2}}$;
3: Perform the condensed SVD of $\mathbf{R}$ as $\mathbf{R} = \mathbf{V}_R\Gamma_R\mathbf{V}_R'$;
4: Calculate $\mathbf{z}$ by (22);
5: Return $\mathbf{z}$ as the q-dimensional representation of $\mathbf{x}$.
6: **end procedure**

Let us investigate the relationship between the solutions of (14) from Algorithms 3 and 5. First, let Υ be obtained from Algorithm 3. It follows from (16) that $(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \mathbf{C}\Upsilon\Lambda$, that is,

$$\mathbf{C}(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{E}\Pi^{-\frac{1}{2}}\Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{C}\Upsilon = \mathbf{C}\Upsilon\Lambda.$$

Thus, $(\Lambda, \Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{C}\Upsilon\Gamma_F^{-1})$ is the nonzero eigenpair of $\mathbf{R}$. Finally, we have $\Lambda = \Gamma_F^2 = \Gamma_R$. In addition, it follows from Theorem 5 that

$$\Gamma_F^{-1}\Upsilon'\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon\Gamma_F^{-1} = \mathbf{I}_q.$$

Moreover, we have

$$\begin{aligned}\tilde{\mathbf{G}}\Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{C}\Upsilon\Gamma_R^{-2} &= \tilde{\mathbf{X}}'\mathbf{H}(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon\Lambda^{-1} \\ &= \tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{C}(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon\Lambda^{-1} \\ &= \tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{C}\Upsilon = \tilde{\mathbf{X}}'\mathbf{H}\Upsilon \end{aligned} \tag{23}$$

because $\tilde{\mathbf{X}}'\mathbf{H} = \tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{C}$. This implies that $\tilde{\mathbf{A}} = \tilde{\mathbf{X}}'\mathbf{H}\Upsilon$ obtained from Algorithm 3 is equivalent to that obtained from Algorithm 5.

On the other hand, let $\mathbf{R} = \mathbf{V}_R\Gamma_R\mathbf{V}_R'$ be the condensed SVD of $\mathbf{R}$. Then

$$\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = (\mathbf{C}^2+\sigma^2\mathbf{C})\Upsilon\Gamma_R,$$

where $\Upsilon = (\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{E}\Pi^{-\frac{1}{2}}\mathbf{V}\Gamma_R^{-\frac{1}{2}}$. Moreover, it is easily checked that

$$\Upsilon'(\mathbf{C}^2+\sigma^2\mathbf{C})\Upsilon = \mathbf{I}_q \quad \text{and} \quad \Upsilon'\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \Gamma_R.$$

This implies that (Γ_R, Υ) is the solution of (16). Again, using (23), we conclude that the solution of (14) from Algorithm 5 is equivalent to the one from Algorithm 3.

In summary, Algorithms 5 and 3 yield equivalent solutions for (14). However, Algorithm 5 is more efficient than Algorithm 3.

6. Relationships Between RFDA and Ridge Regression

It is a well known result that FDA (or KDA) is equivalent to a least mean squared error procedure in the binary classification problem ($c = 2$) (Duda et al., 2001; Mika et al., 2000). Recently, relationships between FDA and a least mean squared error procedure in multi-class ($c > 2$) problems have been discussed by Hastie et al. (2001), Park and Park (2005b), and Ye (2007).

Motivated by this line of work, we investigate a possible equivalency between RFDA and ridge regression (Hoerl and Kennard, 1970). We then go on to consider a similar relationship between RKDA and the corresponding ridge regression problem.

Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]' = \mathbf{E}\Pi^{-\frac{1}{2}}\mathbf{H}_\pi$. That is, $\mathbf{y}_i = (y_{i1}, \ldots, y_{ic})$ is defined by

$$y_{ij} = \begin{cases} \frac{n-n_j}{n\sqrt{n_j}} & \text{if } i \in V_j, \\ -\frac{\sqrt{n_j}}{n} & \text{otherwise.} \end{cases}$$

Regarding $\{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, n\}$ as the training samples, we fit the following multivariate linear function:

$$\mathbf{f}(\mathbf{x}) = \mathbf{w}_0 + \mathbf{W}'\mathbf{x}$$

where $\mathbf{w}_0 \in \mathbb{R}^c$ and $\mathbf{W} \in \mathbb{R}^{p\times c}$. We now find ridge estimates of $\mathbf{w}_0$ and $\mathbf{W}$. In particular, we consider the following minimization problem:

$$\min_{\mathbf{w}_0, \mathbf{W}} L(\mathbf{w}_0, \mathbf{W}) = \frac{1}{2}\|\mathbf{Y} - \mathbf{1}_n\mathbf{w}_0' - \mathbf{X}\mathbf{W}\|_F^2 + \frac{\sigma^2}{2}\mathrm{tr}(\mathbf{W}'\mathbf{W}). \tag{24}$$

We focus on the solution for $\mathbf{W}$:

$$\mathbf{W} = (\mathbf{X}'\mathbf{H}\mathbf{X} + \sigma^2\mathbf{I}_p)^{-1}\mathbf{M}'\Pi^{\frac{1}{2}}\mathbf{H}_\pi = (\mathbf{X}'\mathbf{H}\mathbf{X} + \sigma^2\mathbf{I}_p)^{-1}\mathbf{X}'\mathbf{H}\mathbf{E}\Pi^{-\frac{1}{2}}. \tag{25}$$

The derivation is given in Appendix E. It is then seen from (18) that $\mathbf{W} = \mathbf{G}$. Moreover, when $\sigma^2 = 0$, $\mathbf{W}$ reduces to the ordinary least squares (OLS) estimate of $\mathbf{W}$, which is the solution of the following minimization problem:

$$\min_{\mathbf{w}_0, \mathbf{W}} L(\mathbf{w}_0, \mathbf{W}) = \frac{1}{2}\|\mathbf{Y} - \mathbf{1}_n\mathbf{w}_0' - \mathbf{X}\mathbf{W}\|_F^2. \tag{26}$$

In this case, if $\mathbf{X}'\mathbf{H}\mathbf{X}$ is singular, a standard treatment uses $(\mathbf{X}'\mathbf{H}\mathbf{X})^+$ in (25). Such a $\mathbf{W}$ is identical with $\mathbf{G}$ in (21).

In summary, we have obtained a relationship between the ridge estimation problem in (24) and the RFDA problem in (6).

Theorem 7 *Let* $\mathbf{W}$ *be the solution of the ridge estimation problem in (24) (resp. the OLS estimation problem in (26)) and* $\mathbf{A}$ *be defined in Algorithm 4 for the solution of the RFDA problem in (6) (resp. the FDA problem in (5)). Then*

$$\mathbf{A} = \mathbf{W}\mathbf{V}_R\Gamma_R^{-\frac{1}{2}},$$

where $\mathbf{V}_R$ *and* Γ_R *are defined in Algorithm 4.*

This theorem provides an important connection between $\mathbf{A}$ and $\mathbf{W}$. Indeed $\mathbf{B} = \mathbf{A}\Gamma_R^{\frac{1}{2}}$ is also a solution of the RFDA problem (6). However, $\mathbf{B}$ satisfies the condition $\mathbf{B}'(\mathbf{S}_t + \sigma^2\mathbf{I}_p)\mathbf{B} = \Gamma_R$, rather than $\mathbf{B}'(\mathbf{S}_t + \sigma^2\mathbf{I}_p)\mathbf{B} = \mathbf{I}_q$. Thus, with this $\mathbf{B}$, we obtain the following result, which is the principal theoretical result of this paper.

Theorem 8 *Under the conditions in Theorem 7, we have*

$$\mathbf{B}\mathbf{B}' = \mathbf{W}\mathbf{W}'.$$

Moreover, we have

$$(\mathbf{x}_i - \mathbf{x}_j)'\mathbf{B}\mathbf{B}'(\mathbf{x}_i - \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)'\mathbf{W}\mathbf{W}'(\mathbf{x}_i - \mathbf{x}_j)$$

for any $\mathbf{x}_i$ *and* $\mathbf{x}_j \in \mathbb{R}^p$.

The proof of this theorem is given in Appendix F. Theorem 8 shows that when applying a distance-based classifier such as the K-nearest neighbor (KNN) in the reduced dimensional space, the classification results obtained by the multi-class FDA and multivariate linear estimators are same. Since Theorem 8 holds in general cases, we obtain a complete solution to the open problem concerning the relationship between multi-class FDA problems and multivariate linear estimators.

Similar results have been obtained by Park and Park (2005b); Ye (2007), but under restrictive conditions which arise from a different definition of the label scoring matrix $\mathbf{Y}$ than ours. The choice that of label scoring matrix that we have presented has also been used by Ye (2007), but Ye (2007) attempted to establish a connection between the solution $\mathbf{W}$ and $\mathbf{A}$ as given in Algorithm 1.

It is also worth noting that Zhang and Dai (2009) discussed a connection between the label scoring matrix $\mathbf{Y}$ and the optimal scoring procedure in Hastie et al. (1994). Moreover, Zhang and Jordan (2008) exploited this label scoring matrix in spectral clustering.

Our theorem also goes through immediately in the kernel setting. In particular, for the RKDA problem defined by (11), the corresponding ridge estimator is

$$\min_{\mathbf{w}_0, \mathbf{\Phi} \in \mathbb{R}^{n \times c}} L(\mathbf{w}_0, \Phi) \triangleq \frac{1}{2}\|\mathbf{Y} - \mathbf{1}_n\mathbf{w}_0' - \mathbf{K}\mathbf{H}\Phi\|_F^2 + \frac{\sigma^2}{2}\mathrm{tr}(\Phi'\Phi). \tag{27}$$

The ridge estimation problem corresponding to our RKDA in (14) is given by

$$\min_{\mathbf{w}_0, \tilde{\mathbf{W}} \in \mathbb{R}^{g \times c}} L(\mathbf{w}_0, \tilde{\mathbf{W}}) \triangleq \frac{1}{2}\|\mathbf{Y} - \mathbf{1}_n\mathbf{w}_0' - \tilde{\mathbf{X}}\tilde{\mathbf{W}}\|_F^2 + \frac{\sigma^2}{2}\mathrm{tr}(\tilde{\mathbf{W}}'\tilde{\mathbf{W}}),$$

while the estimation problem for the RKDA in (16) is

$$\min_{\mathbf{w}_0, \mathbf{\Phi} \in \mathbb{R}^{n \times c}} L(\mathbf{w}_0, \Phi) \triangleq \frac{1}{2}\|\mathbf{Y} - \mathbf{1}_n\mathbf{w}_0' - \mathbf{K}\mathbf{H}\Phi\|_F^2 + \frac{\sigma^2}{2}\mathrm{tr}(\Phi'\mathbf{C}\Phi), \tag{28}$$

which is no longer a conventional ridge regression problem. In fact, this problem can be regarded a multi-class extension of the least squares SVM (LS-SVM) (Suykens and Vandewalle, 1999; Suykens et al., 2002); (see, e.g., Van Gestel et al., 2002; Pelckmans et al., 2005). Our work thus provides the relationship between RKDA and the LS-SVM.

Note that when $\sigma^2 = 0$, the problems in (27) and (28) are identical. Moreover, the solution of the problems is given by

$$\Phi = (\mathbf{H}\mathbf{K}\mathbf{H}\mathbf{K}\mathbf{H})^+\mathbf{H}\mathbf{K}\mathbf{H}\mathbf{E}\Pi^{-\frac{1}{2}} = \mathbf{C}^+\mathbf{E}\Pi^{-\frac{1}{2}}.$$

In this case, the RKDA methods in (11) and (16) are also the same. As we see from Section 3, its corresponding pseudoinverse form is given by (15). Let the condensed SVD of $\mathbf{R} = \Pi^{-\frac{1}{2}}\mathbf{E}'\mathbf{C}\mathbf{C}^{+}\mathbf{E}\Pi^{-\frac{1}{2}}$ be $\mathbf{R} = \mathbf{V}_R\Gamma_R\mathbf{V}_R'$. Then $(\Gamma_R, \Phi\mathbf{V}_R\Gamma_R^{-\frac{1}{2}})$ is the solution of (15). This implies that in the case of $\sigma^2 = 0$, there is still the connection between the least squares kernel-based SVM and RKDA shown in Theorem 7.

7. Experimental Study

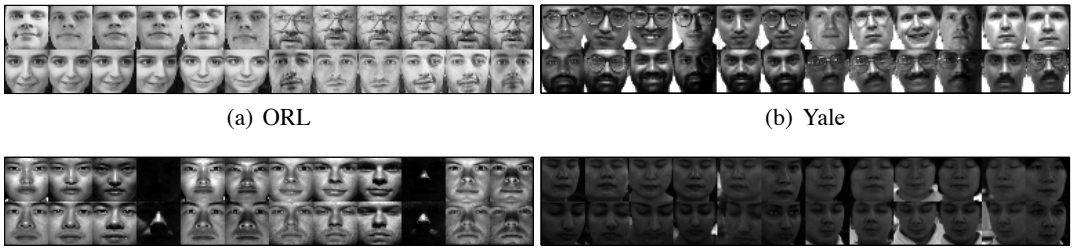
To evaluate the performance of the proposed algorithms for FDA and KDA, we conducted experimental comparisons with other closely related algorithms for FDA and KDA on several real-world data sets. In particular, the comparison was implemented on four face data sets, two handwritten digits data sets, the "letters" data set, and the WebKB data set. All algorithms were implemented in Matlab on a PC configured with an Intel Dual Core 2.0GHz CPU and 2.06GB of memory.

7.1 Setup

The four face data sets are the ORL face database, the Yale face database, the Yale face database B with extension, and the CMU PIE face database, respectively.

- The ORL face database contains 400 facial images of 40 subjects with 10 different images for each subject. This database was developed at the Olivetti Research Laboratories in Cambridge, U.K. The images were taken at different times with variations in facial details (glasses/no glasses), facial expressions (open/closed eyes, smiling/nonsimling), and facial poses (tilted and rotated up to 20 degrees). There is also variation in the scale of up to about 10%. The spatial resolution of the images is 112×92, with 256 gray levels.
- The Yale face images for each subject were captured under different facial expressions or configurations (e.g., center-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, sleepy, surprised, and wink).
- The Yale face database with extension includes the Yale face database B (Georghiades et al., 2001) and the extended Yale Face Database B (Lee et al., 2005). The Yale face database B contains 5760 face images of 10 subjects with 576 different images for each subject, and the extended Yale Face Database B contains 16128 face images of 28 subjects, with each subject having 576 different images. The facial images for each subject were captured under 9 poses and 64 illumination conditions. For the sake of simplicity, a subset called the YaleB&E was collected from two databases; it contains the 2414 face images of 38 subjects.
- The CMU PIE face database contains 41,368 face images of 68 subjects. The facial images for each subject were captured under 13 different poses, 43 different illumination conditions, and with 4 different expressions. In our experiments, we considered only the five near-frontal poses under different illuminations and expressions. For simplicity, we collected a subset of the PIE face database, containing the 6800 face images of 68 subjects with 100 different images for each subject.

In all of the experiments, each whole image was cropped and further resized to have a spatial resolution of 32×32 with 256 gray levels. Figure 1 shows some samples from the four data sets, where four subjects are randomly chosen from each data set and each subject has six sample images.



(a) ORL (b) Yale

(c) YaleB&E (d) PIE

Figure 1: Some sample images randomly chosen from the four data sets, where four subjects from each data set and each subject with six sample images.

The two handwritten digits data sets are the USPS data set and the Binary Alphadigits (BA) data set, respectively.

- The USPS data set was derived from the well-known United States Postal Service (USPS) set of handwritten digits, and contains 2000 images of 10 digits, each digit with 200 images. The spatial resolution of the images in the USPS data set is 16×16, with 256 gray levels.

- The Binary Alphadigits (BA) data set was collected from a binary 20×16 digits database of "0" through "9" and capital "A" through "Z," and thus contains 1404 images of 36 subjects, each subject with 39 image.

The "letters" data set can be obtained from Statlog(`http://www.liacc.up.pt/ML/`) and it consists of images of the letters "A," "B," "C," "D" and "E" with 789, 766, 736, 805 and 768 cases respectively.

Finally, the WebKB data set contains web pages gathered from computer science departments in several universities (Craven et al., 1998). The pages can be divided into seven categories. In our experiments, we used the four most populous categories, namely, student, faculty, course, and project, resulting in a total of 4192 pages. Based on information gain, 300 features were selected.

Table 1 summarizes these benchmark data sets. In our experiments, each data set was randomly partitioned into two disjoint subsets as the training and test data sets, according to the percentage n/k listed in the last column of Table 1. Ten random partitions were obtained for each data set, and several evaluation criteria were reported, including average classification accuracy rate, standard deviation, and average computational time.

The hyperparameters involved in the following methods were selected by cross-validation. After having obtained the q-dimensional representations $\mathbf{z}_i$ of the $\mathbf{x}_i$ from each method, we used a simple nearest neighbor classifier to evaluate the classification accuracy.

7.2 Comparison of FDA Methods

In the linear setting, we compared Algorithm 4 with Algorithm 1 (RFDA/SVD-based), the FDA/GSVD (Howland and Park, 2004) and FDA/MSE methods. Here the FDA/MSE method was derived by Park and Park (2005b) from the relationship between FDA and the minimum squared error solution. We refer

Data set	c	p	k	n/k
ORL	40	1024	400	40%
Yale	15	1024	165	50%
YaleB&E	38	1024	2414	30%
PIE	68	1024	6800	20%
USPS	10	256	2000	10%
BA	36	320	1404	50%
Letters	5	16	3864	10%
WebKB	4	300	4192	10%

Table 1: Summary of the benchmark data sets: $c-$the number of classes; $p-$the dimension of the input vector; $k-$the size of the data set; $n-$the number of the training data.

to Algorithm 4 working with $\mathbf{A}$ as the RFDA/EVD-based method. As we have shown, when $\mathbf{B}$ is used, Algorithm 4 provides the solution of ridge regression. We thus refer to the algorithm working with $\mathbf{B}$ as RFDA/RR. Similar notation also applies to the kernel setting in the next subsection.

Empirically, the performance of the RFDA/EVD-based method is fully identical to that of the RFDA/SVD-based method. This bears out the theoretical analysis in Theorem 6. Thus, we only report the classification accuracies of the RFDA/RR method for Algorithm 4.

Table 2 presents an overall comparison of the methods on all of the data sets and Figure 2 presents the comparative classification results on the four face data sets. It is seen that the RFDA methods have better classification accuracy overall than other methods throughout a range of choices of the number of discriminant variates. Particularly striking is the performance of the RFDA methods when the number of discriminant variates q is small.

From Figure 2, we see that the performance of RFDA/RR method is a little better than that of RFDA/SVD-based method. This implies that RFDA/RR outperforms RFDA/EVD-based method; that is, the performance using the transformation matrix $\mathbf{B}$ is better than that using the transformation matrix $\mathbf{A}$ in Algorithm 4. Therefore, the constraint $\mathbf{A}'(\mathbf{S}_t + \sigma^2\mathbf{I}_p)\mathbf{A} = \mathbf{I}_q$ is not necessarily the best choice for RFDA. This also shows that the ridge regression method given in Section 6 is effective and efficient.

We also compared the computational time of the different methods on the four face data sets. Figure 3 plots the results as a function of the training percentage n/k on the four face data sets. We see that our method has an overall favorable computational complexity in comparison with the other methods on the four face data sets. As the training percentage n/k increases, our method yields more efficient performance.

Note that when the training percentage n/k on the YaleB&E and PIE data sets increases, the singularity problem of the within-class scatter matrix $\mathbf{S}_w$, that is, the small sample size problem, tends to disappear. Figures 3 (c) and (d) show that the FDA/MSE method becomes more efficient in this case, and the corresponding computational time becomes flat with respect to the increase of the training percentage n/k. On the other hand, Figures 3 (c) and (d) also reveal that the computational time of the FDA/SVD-based method significantly increases as the size of training data increases.

Data Set	FDA/GSVD		FDA/MSE		RFDA/SVD-based		RFDA/RR	
	acc ($\pm std$)	*time*	*acc* ($\pm std$)	*time*	*acc* ($\pm std$)	*time*	*acc* ($\pm std$)	*time*
ORL	91.54 ($\pm$1.98)	1.952	91.58 ($\pm$2.00)	0.293	93.17 ($\pm$1.94)	0.347	94.04 ($\pm$1.95)	0.079
Yale	78.56 ($\pm$2.29)	1.281	78.44 ($\pm$2.47)	0.047	79.22 ($\pm$4.19)	0.072	79.56 ($\pm$3.75)	0.014
YaleB&E	59.54 ($\pm$11.8)	43.18	65.34 ($\pm$9.23)	9.967	89.86 ($\pm$1.15)	9.177	90.20 ($\pm$1.09)	1.479
PIE	77.26 ($\pm$1.05)	89.85	77.26 ($\pm$1.05)	23.10	90.40 ($\pm$0.65)	83.88	91.14 ($\pm$0.63)	2.726
USPS	43.02 ($\pm$1.86)	0.392	42.95 ($\pm$1.82)	0.229	82.16 ($\pm$1.07)	0.273	83.49 ($\pm$1.39)	0.035
Letters	91.23 ($\pm$0.98)	0.017	91.23 ($\pm$0.98)	0.011	91.68 ($\pm$0.89)	0.013	91.89 ($\pm$0.65)	0.021
WebKB	67.45 ($\pm$2.29)	0.853	67.45 ($\pm$2.29)	0.595	83.40 ($\pm$0.61)	0.748	83.39 ($\pm$0.63)	0.073
BA	36.40 ($\pm$2.40)	1.586	36.40 ($\pm$2.40)	0.784	68.51 ($\pm$1.91)	0.981	68.85 ($\pm$1.35)	0.157

Table 2: Experimental results for the four methods on different data sets in the linear setting: $acc-$ the best classification accuracy percentage; $std-$ the corresponding standard deviation; $time-$ the corresponding computational time (s).

7.3 Comparison of RKDA Methods

In the kernel setting, we compared Algorithms 2, 3 and RKDA/RR (Algorithm 5 working with **B**). We also implemented the KDA/GSVD method (Park and Park, 2005a) as a baseline. The RBF kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\theta^2)$ was employed, and θ was set to the mean Euclidean distance among training data points. This setting was empirically found to be effective in real-world applications.

Table 3 summarizes the different evaluation criteria on all the data sets. Figures 4 and 5 further illustrate these results. As we see, our two RKDA methods yield better accuracy than the KDA/GSVD method and Algorithm 2. Moreover, RKDA/RR is more efficient computationally than the other methods, especially as the size of training data increases. It should be mentioned here that the data sets in our experiments range from small-sample to large-sample problems. Thus, Table 3 also confirms that the RKDA method based on (14) is more effective and efficient than the method based on (11).

Finally, Figure 6 presents the performance of the four regularized methods with respect to different regularization parameters σ on the four face data sets. From this figure, it can be seen that the regularized parameter σ plays an important role in our RFDA/RR and RKDA/RR methods. Similar results are obtained for the other regularized FDA or KDA methods compared here.

8. Beyond FDA

In this section we extend our results to a more general setting. We first apply the SVD-based algorithm to a family of generalized eigenvector problems, and then propose an efficient algorithm for penalized kernel canonical correlation analysis (KCCA) (Akaho, 2001; Van Gestel et al., 2001; Bach and Jordan, 2002).

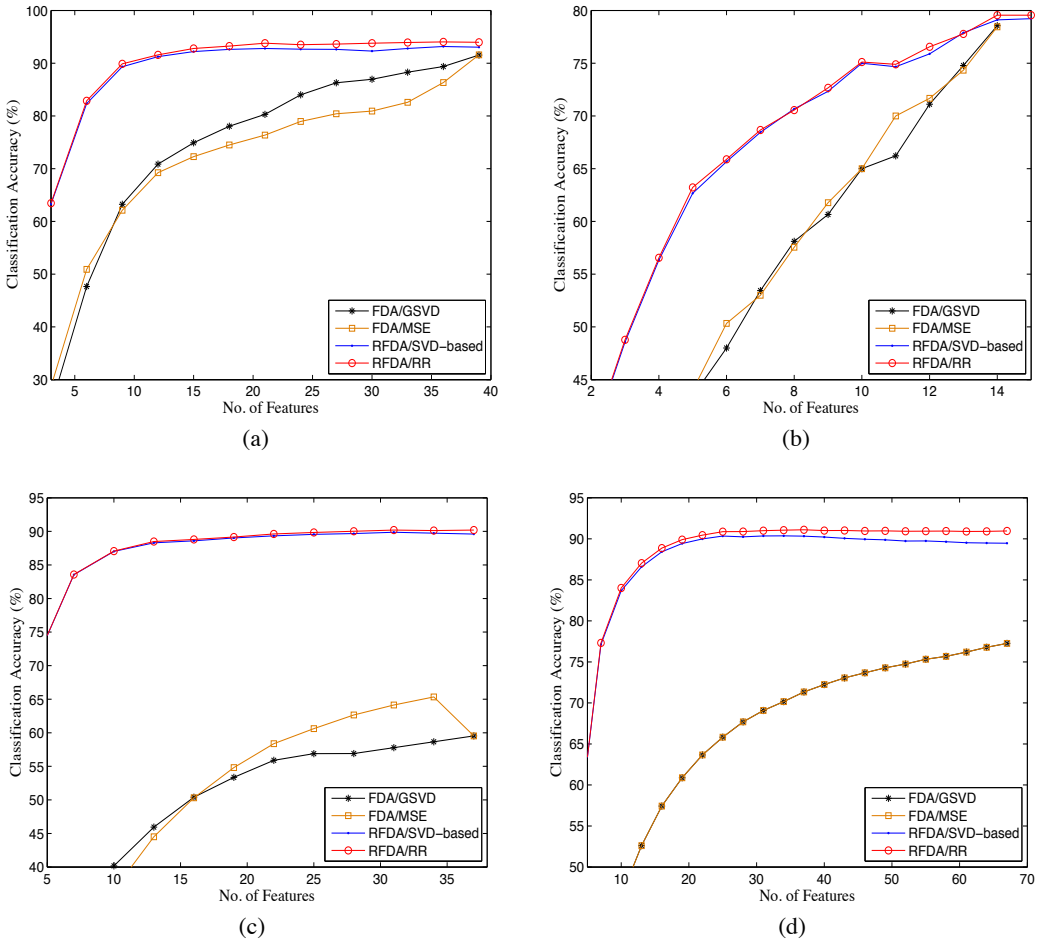


Figure 2: Comparison of the classification accuracies for FDA methods on the four face data sets: (a) ORL; (b) Yale; (c) YaleB&E; (d) PIE.

8.1 A Family of Generalized Eigenvector Problems

Assume that $\mathbf{Q}$ is a $p \times p$ semidefinite positive matrix. Here and later, we define

$$f(\mathbf{Q}) = \sum_{j=0}^{k} b_j \mathbf{Q}^j = b_0 \mathbf{I}_p + b_1 \mathbf{Q} + b_2 \mathbf{Q}^2 + \cdots + b_k \mathbf{Q}^k$$

where $b_0, \ldots, b_k$ are nonnegative real scalars for some positive integer k. We assume that there is at least one b_j such that $b_j > 0$. Let $\mathbf{Q} = \mathbf{V}\Gamma\mathbf{V}'$ be the SVD of $\mathbf{Q}$. We have

$$f(\mathbf{Q}) = \mathbf{V}(b_0 \mathbf{I}_p + b_1 \Gamma + b_2 \Gamma^2 + \cdots + b_k \Gamma^k)\mathbf{V}'.$$

This implies that $f(\mathbf{Q})$ is also semidefinite positive. Moreover, we have $\mathrm{rk}(\mathbf{Q}) \leq \mathrm{rk}(f(\mathbf{Q}))$. In fact, we have $\mathrm{rk}(\mathbf{Q}) = \mathrm{rk}(f(\mathbf{Q}))$ if $b_0 = 0$. However, $f(\mathbf{Q})$ is nonsingular whenever $b_0 > 0$.

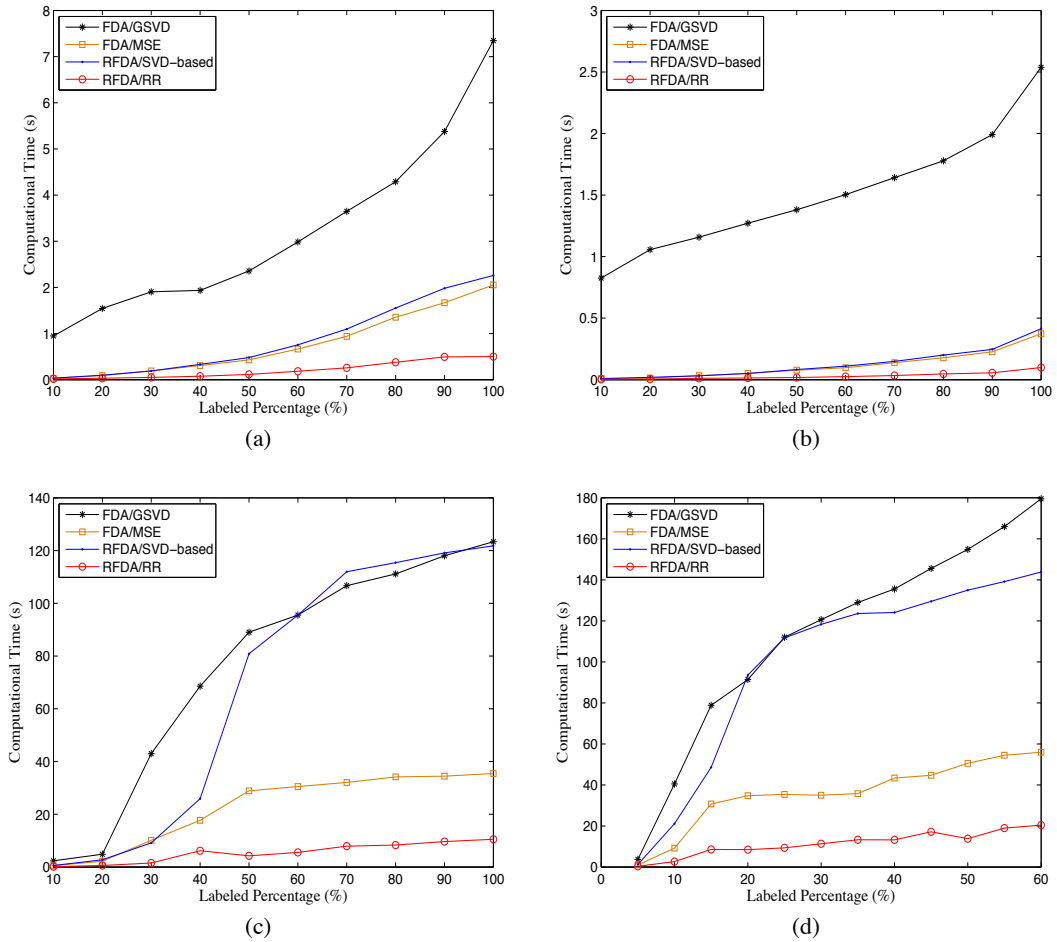


Figure 3: Comparison of the computational times for FDA methods as the training percentage k/n increases on the four face data sets: (a) ORL; (b) Yale; (c) YaleB&E; (d) PIE.

Letting $\mathbf{X} \in \mathbb{R}^{n\times p}$ and $\mathbf{Y} \in \mathbb{R}^{n\times m}$, we consider the following general optimization problem:

$$\max_{\mathbf{A}\in\mathbb{R}^{p\times q}} \operatorname{tr}\big(\mathbf{A}'\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{A}(\mathbf{A}'f(\mathbf{Q})\mathbf{A})^{-1}\big). \tag{29}$$

where $\mathbf{Q} = (\mathbf{X}'\mathbf{X})^{1/2}$ and $\operatorname{rk}(\mathbf{Q}) = \operatorname{rk}(\mathbf{X}) \geq q$. This problem can be formulated as a generalized eigenproblem as follows:

$$\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{A} = f(\mathbf{Q})\mathbf{A}\Lambda. \tag{30}$$

Thus, we consider the following eigenproblem:

$$\big(f(\mathbf{Q})\big)^{+}\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{A} = \mathbf{A}\Lambda. \tag{31}$$

The following theorem shows a relationship between (30) and (31).

Theorem 9 *If $(\Lambda, \mathbf{A})$ (nonzero eigenpairs) is the solution of (31), then $(\Lambda, \mathbf{A})$ is the solution of (30). Conversely, if $(\Lambda, \mathbf{A})$ is the solution of (30), then $(\Lambda, (f(\mathbf{Q}))^{+}f(\mathbf{Q})\mathbf{A})$ is the solution of (31).*

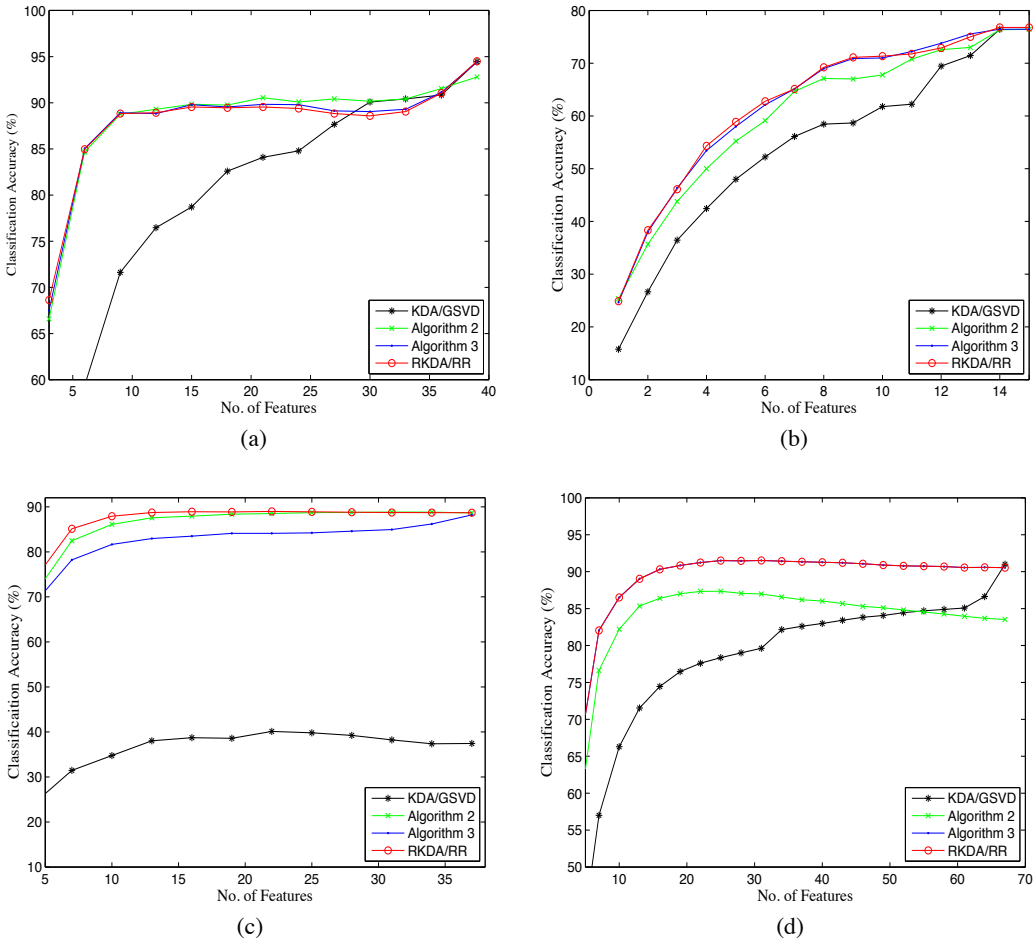


Figure 4: Comparison of the classification accuracies for KDA methods on the four face data sets: (a) ORL; (b) Yale; (c) YaleB&E; (d) PIE.

The theorem obviously holds when $b_0 > 0$, because $f(\mathbf{Q})$ is nonsingular. In the case that $b_0 = 0$, this theorem is a special case of Theorem 1.

Returning to the optimization problem in (29), we have the following theorem.

Theorem 10 *Assume that* $\mathrm{rk}(\mathbf{X}) = r \geq q$. *Let the condensed SVD of* $\mathbf{X}$ *be* $\mathbf{X} = \mathbf{U}_X \Gamma_X \mathbf{V}_X'$ *and the condensed SVD of* $\mathbf{F} = (f(\Gamma_X))^{-\frac{1}{2}} \Gamma_X \mathbf{U}_X' \mathbf{Y}$ *be* $\mathbf{F} = \mathbf{U}_F \Gamma_F \mathbf{V}_F'$. *We have: (i)* $(\Lambda, \mathbf{T})$ *where* $\mathbf{T} = \mathbf{V}_X (f(\Gamma_X))^{-\frac{1}{2}} \mathbf{U}_F$ *and* $\Lambda = \Gamma_F^2$ *are* r *eigenpairs of the pencil* $(\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}, f(\mathbf{Q}))$; *(ii) the matrix* $\mathbf{T}_q$ *consisting of the first* q *columns of* $\mathbf{T}$ *is a maximizer of the generalized Rayleigh quotient in* (29).

The proof is given in Appendix G. This theorem shows that we can use the SVD-based algorithm to solve (29). That is, we obtain a derivation of Algorithm 6. Moreover, it is easily seen that the RFDA problems (6), (11), (14) and (16) are special cases of the problem (29) with different settings for the b_j.

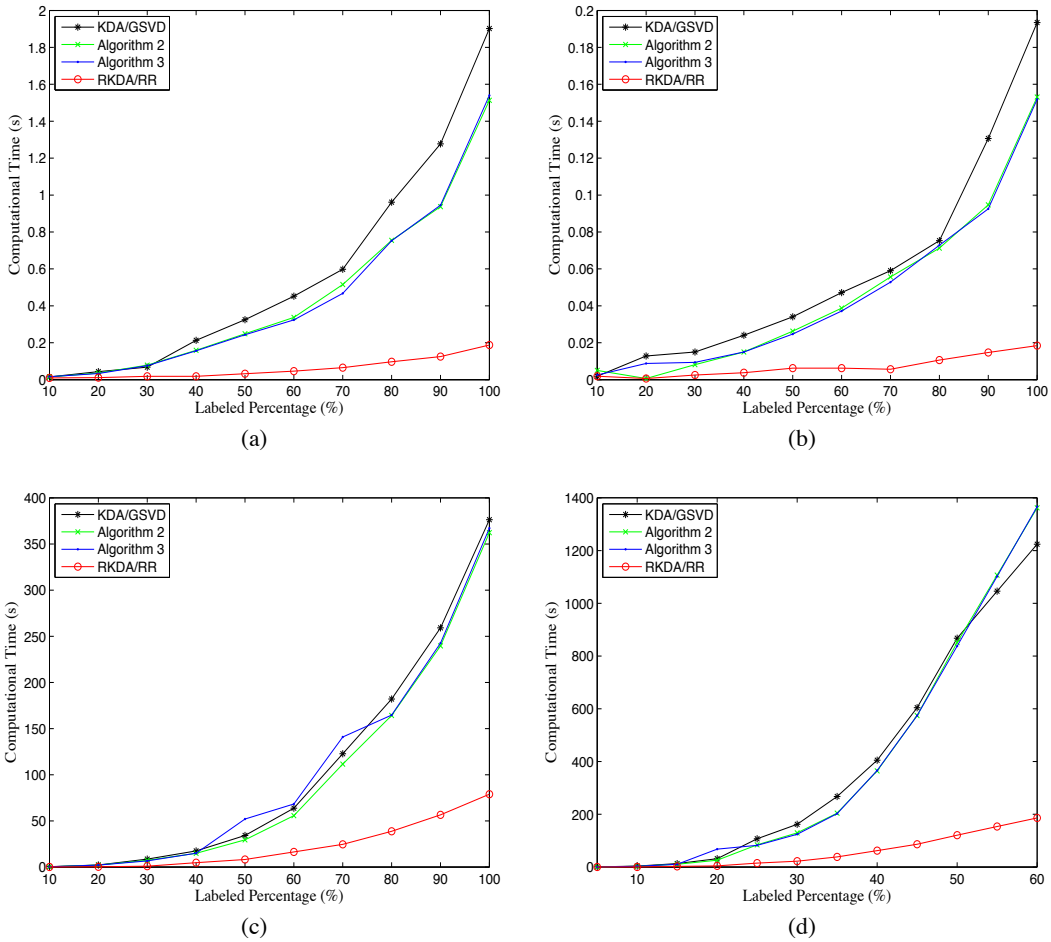


Figure 5: Comparison of the computational times for KDA methods as the training percentage k/n increases on the four face data sets: (a) ORL; (b) Yale; (c) YaleB&E; (d) PIE.

Algorithm 6 SVD-based Algorithm for Problem (29)

1: **procedure** GEP($\{\mathbf{X}, \mathbf{Y}, \sigma^2\}$)
2: Perform the condensed SVD of $\mathbf{X}$ as $\mathbf{X} = \mathbf{U}_X \Gamma_X \mathbf{V}_X'$.
3: Calculate $\mathbf{F} = (f(\Gamma_X))^{-\frac{1}{2}} \Gamma_X \mathbf{U}_X' \mathbf{Y}$ where $r = \mathrm{rk}(\Gamma_X)$.
4: Perform the condensed SVD of $\mathbf{F}$ as $\mathbf{F} = \mathbf{U}_F \Gamma_F \mathbf{V}_F'$.
5: Let $\mathbf{T} = \mathbf{V}_X (f(\Gamma_X))^{-\frac{1}{2}} \mathbf{U}_F$
6: Return $\mathbf{A} = \mathbf{T}(:, 1:q)$ for $q \leq r$ as a maximizer of Problem (29).
7: **end procedure**

8.2 Penalized KCCA

Given two data matrices $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{n \times p}$ and $\mathbf{Y} \in \mathcal{Y} \subset \mathbb{R}^{n \times m}$, CCA finds two matrices $\mathbf{A}_x \in \mathbb{R}^{p \times q}$ and $\mathbf{A}_y \in \mathbb{R}^{m \times q}$ of canonical correlation vectors by solving the following optimization problem:

$$\max_{\mathbf{A}_x, \mathbf{A}_y} \mathrm{tr}(\mathbf{A}_x' \mathbf{S}_{xy} \mathbf{A}_y),$$

$$\text{s.t. } \mathbf{A}_x' \mathbf{S}_{xx} \mathbf{A}_x = \mathbf{I}_q \text{ and } \mathbf{A}_y' \mathbf{S}_{yy} \mathbf{A}_y = \mathbf{I}_q,$$

Data Set	KDA/GSVD		Algorithm 2		Algorithm 3		RKDA/RR	
	acc ($\pm std$)	*time*	*acc* ($\pm std$)	*time*	*acc* ($\pm std$)	*time*	*acc* ($\pm std$)	*time*
ORL	94.45 ($\pm$1.63)	0.231	92.79 ($\pm$1.74)	0.159	94.41 ($\pm$2.03)	0.162	94.50 ($\pm$1.64)	0.032
Yale	76.44 ($\pm$3.50)	0.017	76.44 ($\pm$2.71)	0.025	76.44 ($\pm$2.38)	0.025	76.78 ($\pm$3.20)	0.004
YaleB&E	40.34 ($\pm$22.4)	7.898	88.83 ($\pm$0.99)	6.520	88.20 ($\pm$0.88)	6.554	89.06 ($\pm$0.81)	0.818
PIE	91.00 ($\pm$0.36)	48.07	87.33 ($\pm$0.65)	40.71	91.52 ($\pm$0.45)	40.90	91.52 ($\pm$0.45)	5.079
USPS	82.25 ($\pm$1.59)	0.305	83.96 ($\pm$1.10)	0.238	84.92 ($\pm$1.56)	0.234	83.94 ($\pm$0.84)	0.020
Letters	92.74 ($\pm$2.10)	1.162	95.88 ($\pm$0.63)	1.100	94.60 ($\pm$0.99)	1.028	96.05 ($\pm$0.67)	0.134
WebKB	77.27 ($\pm$2.77)	1.684	83.51 ($\pm$0.51)	1.464	83.47 ($\pm$0.49)	1.452	83.47 ($\pm$0.49)	0.156
BA	66.19 ($\pm$1.21)	7.883	68.70 ($\pm$1.76)	6.715	69.86 ($\pm$1.30)	6.626	69.82 ($\pm$1.10)	0.709

Table 3: Experimental results for the five methods on different data sets in the kernel setting: *acc*– the best classification accuracy percentage; *std*– the corresponding standard deviation; *time*– the corresponding computational time (*s*).

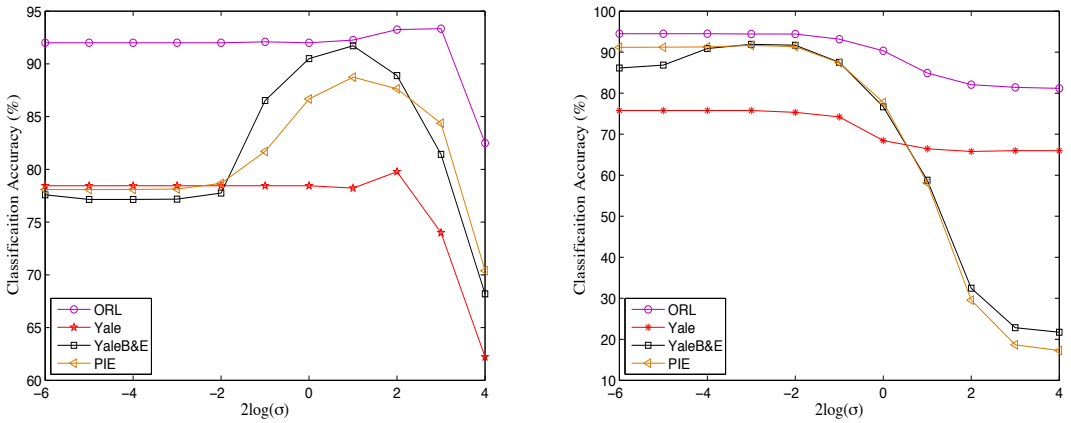


Figure 6: Performance of the RFDA/RR and RKDA/RR methods for different regularization parameters σ, where (a) displays the results of RFDA/RR on different data sets and (b) displays the results of RKDA/RR on different data sets.

where $q \leq \min\{p, m, n-1\}$, $\mathbf{S}_{xx} = \mathbf{X}'\mathbf{H}\mathbf{X}$ and $\mathbf{S}_{yy} = \mathbf{Y}'\mathbf{H}\mathbf{Y}$ are the pooled covariance matrices of $\mathbf{x}$ and $\mathbf{y}$, respectively, and $\mathbf{S}_{xy} = \mathbf{X}'\mathbf{H}\mathbf{Y} = \mathbf{S}'_{yx}$ is the pooled cross-covariance matrix between $\mathbf{x}$ and $\mathbf{y}$.

Consider that either $\mathbf{S}_{xx}$ or $\mathbf{S}_{yy}$ is ill-conditioned. The penalized CCA method (Hastie et al., 1995) solves the following optimization problem

$$\max_{\mathbf{A}_x, \mathbf{A}_y} \ \mathrm{tr}(\mathbf{A}'_x \mathbf{S}_{xy} \mathbf{A}_y),$$

$$\text{s.t. } \mathbf{A}'_x(\mathbf{S}_{xx} + \sigma_x^2 \mathbf{I}_p)\mathbf{A}_x = \mathbf{I}_q \ \text{ and } \ \mathbf{A}'_y(\mathbf{S}_{yy} + \sigma_y^2 \mathbf{I}_m)\mathbf{A}_y = \mathbf{I}_q.$$

This problem can be solved in a two-step process (Mardia et al., 1979). The first step solves the following generalized problem:

$$\mathbf{S}_{yx}(\mathbf{S}_{xx}+\sigma_x^2\mathbf{I}_p)^{-1}\mathbf{S}_{xy}\mathbf{A}_y = (\mathbf{S}_{yy}+\sigma_y^2\mathbf{I}_m)\mathbf{A}_y\Lambda,$$

where Λ is a $q\times q$ diagonal matrix with positive diagonal elements. The second step calculates $\mathbf{A}_x$ by

$$\mathbf{A}_x = (\mathbf{S}_{xx}+\sigma_x^2\mathbf{I}_p)^{-1}\mathbf{S}_{xy}\mathbf{A}_y\Lambda^{-\frac{1}{2}}.$$

Assume that we have kernel functions $K_x(\cdot,\cdot)$: $\mathcal{X}\times\mathcal{X}\to\mathbb{R}$, and $K_y(\cdot,\cdot)$: $\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$. Similar to the linear case, penalized KCCA first solves the following generalized problem:

$$\tilde{\mathbf{S}}_{xy}(\tilde{\mathbf{S}}_{yy}+\sigma_y^2\mathbf{I}_h)^{-1}\tilde{\mathbf{S}}_{yx}\tilde{\mathbf{A}}_x = (\tilde{\mathbf{S}}_{xx}+\sigma_x^2\mathbf{I}_g)\tilde{\mathbf{A}}_x\Lambda, \tag{32}$$

and then calculates $\tilde{\mathbf{A}}_y$ by

$$\tilde{\mathbf{A}}_y = (\tilde{\mathbf{S}}_{yy}+\sigma_y^2\mathbf{I}_h)^{-1}\tilde{\mathbf{S}}_{yx}\tilde{\mathbf{A}}_x\Lambda^{-\frac{1}{2}}.$$

Here g and h are the dimensions of the corresponding feature spaces. We now address the solution to the generalized eigenproblem in (32). Consider

$$\begin{aligned}
&(\tilde{\mathbf{S}}_{xx}+\sigma_x^2\mathbf{I}_g)^{-1}\tilde{\mathbf{S}}_{xy}(\tilde{\mathbf{S}}_{yy}+\sigma_y^2\mathbf{I}_h)^{-1}\tilde{\mathbf{S}}_{yx}\\
&= (\tilde{\mathbf{X}}'\mathbf{H}\tilde{\mathbf{X}}+\sigma_x^2\mathbf{I}_g)^{-1}\tilde{\mathbf{X}}'\mathbf{H}\tilde{\mathbf{Y}}(\tilde{\mathbf{Y}}'\mathbf{H}\tilde{\mathbf{Y}}+\sigma_y^2\mathbf{I}_h)^{-1}\tilde{\mathbf{Y}}'\mathbf{H}\tilde{\mathbf{X}}\\
&= \tilde{\mathbf{X}}'\mathbf{H}(\mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H}+\sigma_x^2\mathbf{I}_n)^{-1}(\mathbf{H}\tilde{\mathbf{Y}}\tilde{\mathbf{Y}}'\mathbf{H}+\sigma_y^2\mathbf{I}_n)^{-1}\mathbf{H}\tilde{\mathbf{Y}}\tilde{\mathbf{Y}}'\mathbf{H}\tilde{\mathbf{X}}\\
&= \tilde{\mathbf{X}}'\mathbf{H}(\mathbf{C}_x+\sigma_x^2\mathbf{I}_n)^{-1}(\mathbf{C}_y+\sigma_y^2\mathbf{I}_n)^{-1}\mathbf{C}_y\mathbf{H}\tilde{\mathbf{X}},
\end{aligned}$$

where $\mathbf{C}_x = \mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H}$ and $\mathbf{C}_y = \mathbf{H}\tilde{\mathbf{Y}}\tilde{\mathbf{Y}}'\mathbf{H}$. Since $\tilde{\mathbf{X}}'\mathbf{H}(\mathbf{C}_x+\sigma_x^2\mathbf{I}_n)^{-1}(\mathbf{C}_y+\sigma_y^2\mathbf{I}_n)^{-1}\mathbf{C}_y\mathbf{H}\tilde{\mathbf{X}}$ and $(\mathbf{C}_x+\sigma_x^2\mathbf{I}_n)^{-1}(\mathbf{C}_y+\sigma_y^2\mathbf{I}_n)^{-1}\mathbf{C}_y\mathbf{C}_x$ have the same nonzero eigenvalues, we let

$$(\mathbf{C}_x+\sigma_x^2\mathbf{I}_n)^{-1}(\mathbf{C}_y+\sigma_y^2\mathbf{I}_n)^{-1}\mathbf{C}_y\mathbf{C}_x\Upsilon = \Upsilon\Lambda$$

where Λ consists of the q largest nonzero eigenvalues of $(\mathbf{C}_x+\sigma_x^2\mathbf{I}_n)^{-1}(\mathbf{C}_y+\sigma_y^2\mathbf{I}_n)^{-1}\mathbf{C}_y\mathbf{C}_x$. We thus define

$$\tilde{\mathbf{A}}_x = \tilde{\mathbf{X}}'\mathbf{H}\Upsilon$$

and

$$\tilde{\mathbf{A}}_y = \tilde{\mathbf{Y}}'\mathbf{H}(\mathbf{C}_y+\sigma_y^2\mathbf{I}_n)^{-1}\mathbf{C}_x\Upsilon\Lambda^{-\frac{1}{2}}$$

as the solution of the KCCA problem. Given $\mathbf{x}\in\mathbb{R}^p$ and $\mathbf{y}\in\mathbb{R}^m$, we can directly calculate their canonical variables by

$$\mathbf{z}_x = \tilde{\mathbf{A}}_x'(\tilde{\mathbf{x}}-\tilde{\mathbf{m}}_x) = \Upsilon'\Big(\mathbf{k}_x-\frac{1}{n}\mathbf{K}_x\mathbf{1}_n\Big),$$

where $\tilde{\mathbf{m}}_x = \frac{1}{n}\sum_{i=1}^n\tilde{\mathbf{x}}_i$, $\mathbf{k}_x = (K_x(\mathbf{x},\mathbf{x}_1),\ldots,K_x(\mathbf{x},\mathbf{x}_n))'$ and $\mathbf{K}_x = \tilde{\mathbf{X}}\tilde{\mathbf{X}}'$, and

$$\mathbf{z}_y = \tilde{\mathbf{A}}_y'(\tilde{\mathbf{y}}-\tilde{\mathbf{m}}_y) = \Lambda^{-\frac{1}{2}}\Upsilon'\mathbf{C}_x(\mathbf{C}_y+\sigma_y^2\mathbf{I}_n)^{-1}\Big(\mathbf{k}_y-\frac{1}{n}\mathbf{K}_y\mathbf{1}_n\Big),$$

where $\tilde{\mathbf{m}}_y = \frac{1}{n}\sum_{i=1}^n\tilde{\mathbf{y}}_i$, $\mathbf{k}_y = (K_y(\mathbf{y},\mathbf{y}_1),\ldots,K_y(\mathbf{y},\mathbf{y}_n))'$ and $\mathbf{K}_y = \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}'$.

As we see, the canonical vectors $\mathbf{z}_x$ and $\mathbf{z}_y$ can be calculated without the explicit use of $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$. Moreover, if $\sigma_x^2=0$ or $\sigma_y^2=0$, the algorithm still works by using $\mathbf{C}_x^+$ or $\mathbf{C}_y^+$ instead.

9. Conclusion

In this paper we have provided a solution to an open problem concerning the relationship between multi-class discriminant analysis problems and multivariate regression problems, both in the linear setting and the kernel setting. Our theory has yielded efficient and effective algorithms for FDA and KDA within both the regularization and pseudoinverse paradigms. The favorable performance of our algorithms has been demonstrated empirically on a collection of benchmark data sets. We have also extended our algorithms to a more general family of generalized eigenvalue problems.

Acknowledgments

Zhihua Zhang and Congfu Xu acknowledge support from the 973 Program of China (No. 2010CB327903). Zhihua Zhang acknowledges support from Natural Science Foundations of China (No. 61070239), Doctoral Program of Specialized Research Fund of Chinese Universities, and the Fundamental Research Funds for the Central Universities. Michael Jordan acknowledges support from Google, Intel and Microsoft Research.

Appendix A. Proof of Theorem 1

Let $\Sigma_1 = \mathbf{U}_1\Gamma_1\mathbf{V}_1'$ and $\Sigma_2 = \mathbf{U}_2\Gamma_2\mathbf{V}_2'$ be the condensed SVD of Σ_1 and Σ_2. Thus, we have $\mathcal{R}(\Sigma_1) = \mathcal{R}(\mathbf{U}_1)$ and $\mathcal{R}(\Sigma_2) = \mathcal{R}(\mathbf{U}_2)$. Moreover, we have $\Sigma_2^+ = \mathbf{V}_2\Gamma_2^{-1}\mathbf{U}_2'$ and $\Sigma_2\Sigma_2^+ = \mathbf{U}_2\mathbf{U}_2'$. It follows from $\mathcal{R}(\Sigma_1) \subseteq \mathcal{R}(\Sigma_2)$ that $\mathcal{R}(\mathbf{U}_1) \subseteq \mathcal{R}(\mathbf{U}_2)$. This implies that $\mathbf{U}_1$ can be expressed as $\mathbf{U}_1 = \mathbf{U}_2\mathbf{Q}$ where $\mathbf{Q}$ is some matrix of appropriate order. As a result, we have

$$\Sigma_2\Sigma_2^+\Sigma_1 = \mathbf{U}_2\mathbf{U}_2'\mathbf{U}_2\mathbf{Q}\Gamma_1\mathbf{V}_1' = \Sigma_1.$$

It is worth noting that the condition $\Sigma_2\Sigma_2^+\Sigma_1 = \Sigma_1$ is not only necessary but also sufficient for $\mathcal{R}(\Sigma_1) \subseteq \mathcal{R}(\Sigma_2)$.

If $(\Lambda, \mathbf{B})$ are the eigenpairs of $\Sigma_2^+\Sigma_1$, then it is easily seen that $(\Lambda, \mathbf{B})$ are also the eigenpairs of (Σ_1, Σ_2) due to $\Sigma_2\Sigma_2^+\Sigma_1 = \Sigma_1$.

Conversely, suppose $(\Lambda, \mathbf{B})$ are the eigenpairs of (Σ_1, Σ_2). Then we have $\Sigma_2\Sigma_2^+\Sigma_1\mathbf{B} = \Sigma_2\mathbf{B}\Lambda$. This implies that $(\Lambda, \Sigma_2^+\Sigma_2\mathbf{B})$ are the eigenpairs of $\Sigma_2^+\Sigma_1$ due to $\Sigma_2\Sigma_2^+\Sigma_1 = \Sigma_1$ and $\Sigma_2^+\Sigma_2\Sigma_2^+ = \Sigma_2^+$.

Appendix B. Some Properties of Moore-Penrose Inverses

In order to prove Theorem 3, we will need some properties of Moore-Penrose inverses.

Lemma 11 *Let* $\mathbf{C} = \mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H}$, $\tilde{\mathbf{S}}_t = \tilde{\mathbf{X}}'\mathbf{H}\tilde{\mathbf{X}}$ *and* $\tilde{\mathbf{S}}_b = \tilde{\mathbf{X}}'\mathbf{H}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{H}\tilde{\mathbf{X}}$. *Then*

(a) $\mathbf{C}\mathbf{C}^+ = (\mathbf{C}\mathbf{C}^+)' = \mathbf{C}^+\mathbf{C}$, $\mathbf{C}^+\mathbf{C}\mathbf{C} = \mathbf{C}\mathbf{C}^+\mathbf{C} = \mathbf{C}$, $\mathbf{C}^+\mathbf{C}^+\mathbf{C} = \mathbf{C}^+\mathbf{C}\mathbf{C}^+ = \mathbf{C}^+$;

(b) $\tilde{\mathbf{S}}_t^+\tilde{\mathbf{X}}'\mathbf{H} = (\tilde{\mathbf{X}}'\mathbf{H}\mathbf{H}\tilde{\mathbf{X}})^+\tilde{\mathbf{X}}'\mathbf{H} = \tilde{\mathbf{X}}'\mathbf{H}(\mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H})^+ = \tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+$;

(c) $\tilde{\mathbf{X}}'\mathbf{H} = \tilde{\mathbf{S}}_t\tilde{\mathbf{S}}_t^+\tilde{\mathbf{X}}'\mathbf{H} = \tilde{\mathbf{X}}'\mathbf{H}\mathbf{H}\tilde{\mathbf{X}}(\tilde{\mathbf{X}}'\mathbf{H}\mathbf{H}\tilde{\mathbf{X}})^+\tilde{\mathbf{X}}'\mathbf{H}$
$= \tilde{\mathbf{X}}'\mathbf{H}(\mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H})^+\mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H} = \tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{C}$;

(d) $\tilde{\mathbf{S}}_t\tilde{\mathbf{S}}_t^+\tilde{\mathbf{S}}_b = \tilde{\mathbf{S}}_b$.

These results can be found in Lütkepohl (1996).

Appendix C. Proof of Theorem 3

First, if (Λ, Υ) is the solution of (10), we have

$$\begin{aligned}\tilde{\mathbf{S}}_b\tilde{\mathbf{X}}'\mathbf{H}\Upsilon &= \tilde{\mathbf{X}}'\mathbf{H}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon \\ &= \tilde{\mathbf{X}}'\mathbf{H}\mathbf{H}\tilde{\mathbf{X}}(\tilde{\mathbf{X}}'\mathbf{H}\mathbf{H}\tilde{\mathbf{X}})^+\tilde{\mathbf{X}}'\mathbf{H}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon \\ &= \tilde{\mathbf{S}}_t\tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \tilde{\mathbf{S}}_t\tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{C}^+\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon \\ &= \tilde{\mathbf{S}}_t\tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{C}^+\mathbf{C}\mathbf{C}\Upsilon\Lambda = \tilde{\mathbf{S}}_t\tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{C}\Upsilon\Lambda \\ &= \tilde{\mathbf{S}}_t\tilde{\mathbf{X}}'\mathbf{H}\Upsilon\Lambda.\end{aligned}$$

This implies that $\left(\Lambda, \tilde{\mathbf{X}}'\mathbf{H}\Upsilon\right)$ is the solution of (7).

Second, if (Λ, Υ) is the solution of (15), we have

$$\begin{aligned}\tilde{\mathbf{S}}_t^+\tilde{\mathbf{S}}_b\tilde{\mathbf{X}}'\mathbf{H}\Upsilon &= (\tilde{\mathbf{X}}'\mathbf{H}\mathbf{H}\tilde{\mathbf{X}})^+\tilde{\mathbf{X}}'\mathbf{H}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{H}\tilde{\mathbf{X}}\tilde{\mathbf{X}}'\mathbf{H}\Upsilon \\ &= \tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon \\ &= \tilde{\mathbf{X}}'\mathbf{H}\Upsilon\Lambda.\end{aligned}$$

This implies that $(\Lambda, \tilde{\mathbf{X}}'\mathbf{H}\Upsilon)$ is the solution of (13).

Finally, it follows from (16) that

$$(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \mathbf{C}\Upsilon\Lambda.$$

In addition, note that $(\tilde{\mathbf{S}}_t+\sigma^2\mathbf{I}_g)^{-1}\tilde{\mathbf{X}}'\mathbf{H} = \tilde{\mathbf{X}}'\mathbf{H}(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}$. Hence, we have

$$\begin{aligned}(\tilde{\mathbf{S}}_t+\sigma^2\mathbf{I}_g)^{-1}\tilde{\mathbf{S}}_b\tilde{\mathbf{X}}'\mathbf{H}\Upsilon &= (\tilde{\mathbf{S}}_t+\sigma^2\mathbf{I}_g)^{-1}\tilde{\mathbf{X}}'\mathbf{H}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon \\ &= \tilde{\mathbf{X}}'\mathbf{H}(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon \\ &= \tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{C}(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon \\ &= \tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+(\mathbf{C}+\sigma^2\mathbf{I}_n)^{-1}\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \tilde{\mathbf{X}}'\mathbf{H}\mathbf{C}^+\mathbf{C}\Upsilon \\ &= \tilde{\mathbf{X}}'\mathbf{H}\Upsilon\Lambda.\end{aligned}$$

This completes the proof of part (iii).

Appendix D. Proof of Theorem 5

We prove the final part. As for other parts, their proof can be immediately obtained from Appendix G. In terms of Algorithm 3, we have

$$\begin{aligned}\tilde{\mathbf{A}}'(\tilde{\mathbf{S}}_t+\sigma^2\mathbf{I}_g)\tilde{\mathbf{A}} &= \Upsilon'\mathbf{H}\tilde{\mathbf{X}}(\tilde{\mathbf{S}}_t+\sigma^2\mathbf{I}_g)\tilde{\mathbf{X}}'\mathbf{H}\Upsilon \\ &= \Upsilon'\mathbf{C}(\mathbf{C}+\sigma^2\mathbf{I}_n)\Upsilon = \mathbf{I}_q\end{aligned}$$

and

$$\tilde{\mathbf{A}}'\tilde{\mathbf{S}}_b\tilde{\mathbf{A}} = \Upsilon'\mathbf{C}\mathbf{E}\Pi^{-1}\mathbf{E}'\mathbf{C}\Upsilon = \Gamma_F^2.$$

Appendix E. Derivation of Equation 25

The first-order derivatives of $L(\mathbf{w}_0, \mathbf{W})$ with respect to $\mathbf{w}_0$ and $\mathbf{W}$ are given by

$$\frac{\partial L}{\partial \mathbf{w}_0} = n\mathbf{w}_0 + \mathbf{W}'\mathbf{X}'\mathbf{1}_n - \mathbf{Y}'\mathbf{1}_n,$$
$$\frac{\partial L}{\partial \mathbf{W}} = (\mathbf{X}'\mathbf{X} + \sigma^2\mathbf{I}_p)\mathbf{W} + \mathbf{X}'\mathbf{1}_n\mathbf{w}_0' - \mathbf{X}'\mathbf{Y},$$

Letting $\frac{\partial L}{\partial \mathbf{w}_0} = \mathbf{0}$, $\frac{\partial L}{\partial \mathbf{W}} = \mathbf{0}$ and $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i = \frac{1}{n}\mathbf{X}'\mathbf{1}_n$, we obtain

$$\begin{cases} \mathbf{w}_0 + \mathbf{W}'\bar{\mathbf{x}} = \mathbf{0} \\ n\bar{\mathbf{x}}\mathbf{w}_0' + (\mathbf{X}'\mathbf{X} + \sigma^2\mathbf{I}_p)\mathbf{W} = \mathbf{M}'\Pi^{\frac{1}{2}}\mathbf{H}_\pi \end{cases}$$

due to $\mathbf{Y}'\mathbf{1}_n = \mathbf{0}$ and $\mathbf{X}'\mathbf{Y} = \mathbf{M}'\Pi^{\frac{1}{2}}\mathbf{H}_\pi$. Further, it follows that $\mathbf{w}_0 = -\mathbf{W}\bar{\mathbf{x}}$, and hence,

$$(\mathbf{X}'\mathbf{H}\mathbf{X} + \sigma^2\mathbf{I}_p)\mathbf{W} = \mathbf{M}'\Pi^{\frac{1}{2}}\mathbf{H}_\pi$$

because of $\mathbf{X}'\mathbf{X} - n\bar{\mathbf{x}}\bar{\mathbf{x}}' = \mathbf{X}'\mathbf{H}\mathbf{X}$. We thus obtain $\mathbf{W}$ in (25). It then follows from (18) that $\mathbf{W} = \mathbf{G}$. Moreover, when $\sigma^2 = 0$, $\mathbf{W}$ reduces to the solution of the minimization problem in (26). In this case, if $\mathbf{X}'\mathbf{H}\mathbf{X}$ is singular, a standard treatment is to use the Moore-Penrose inverse $(\mathbf{X}'\mathbf{H}\mathbf{X})^+$ in (25). Such a $\mathbf{W}$ is identical with $\mathbf{G}$ in (21).

Appendix F. Proof of Theorem 8

Since $\mathbf{V}_R$ is an $c\times q$ orthogonal matrix, there exists a $c\times(c-q)$ orthogonal matrix $\mathbf{V}_2$ such that $\mathbf{V} = [\mathbf{V}_R, \mathbf{V}_2]$ is a $c\times c$ orthogonal matrix. Noting that $\mathbf{R} = \mathbf{V}_R\Gamma_R\mathbf{V}_R'$, we have $\mathbf{R}\mathbf{V}_2 = \mathbf{0}$ and $\mathbf{V}_2'\mathbf{R}\mathbf{V}_2 = \mathbf{0}$. Let $\mathbf{Q} = \mathbf{M}'\Pi^{\frac{1}{2}}\mathbf{H}_\pi\mathbf{V}_2$. Then we obtain $\mathbf{Q}'(\mathbf{X}'\mathbf{H}\mathbf{X}+\sigma^2\mathbf{I}_p)^{-1}\mathbf{Q} = \mathbf{0}$. This implies $\mathbf{Q} = \mathbf{0}$ because $(\mathbf{X}'\mathbf{H}\mathbf{X} + \sigma^2\mathbf{I}_p)^{-1}$ is positive definite. Hence, $\mathbf{W}\mathbf{V}_2 = (\mathbf{X}'\mathbf{H}\mathbf{X} + \sigma^2\mathbf{I}_p)^{-1}\mathbf{Q} = \mathbf{0}$. As a result, we have

$$\begin{aligned} \mathbf{W}\mathbf{W}' &= \mathbf{W}\mathbf{V}\mathbf{V}'\mathbf{W}' \\ &= \mathbf{W}\mathbf{V}_R\mathbf{V}_R'\mathbf{W}' + \mathbf{W}\mathbf{V}_2\mathbf{V}_2'\mathbf{W}' \\ &= \mathbf{B}\mathbf{B}'. \end{aligned}$$

Note that if $\sigma^2 = 0$ and $\mathbf{X}'\mathbf{H}\mathbf{X}$ is nonsingular, we still have $\mathbf{W}\mathbf{W}' = \mathbf{B}\mathbf{B}'$. In the case that $\mathbf{X}'\mathbf{H}\mathbf{X}$ is singular, we have $\mathbf{Q}'(\mathbf{X}'\mathbf{H}\mathbf{X})^+\mathbf{Q} = \mathbf{0}$. Since $(\mathbf{X}'\mathbf{H}\mathbf{X})^+$ is positive semidefinite, its square root matrix exists and it is denoted by Ω. It thus follows from $\mathbf{Q}'(\mathbf{X}'\mathbf{H}\mathbf{X})^+\mathbf{Q} = \mathbf{Q}\Omega\Omega\mathbf{Q}' = \mathbf{0}$ that $\Omega\mathbf{Q}' = \mathbf{0}$. This shows that $\mathbf{W}\mathbf{V}_2 = (\mathbf{X}'\mathbf{H}\mathbf{X})^+\mathbf{Q} = \mathbf{0}$. Thus, we also obtain $\mathbf{W}\mathbf{W}' = \mathbf{B}\mathbf{B}'$. The proof is complete.

Appendix G. Proof of Theorem 10

It is immediate that

$$\begin{aligned} \mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{T} &= \mathbf{V}_X\Gamma_X\mathbf{U}_X'\mathbf{Y}\mathbf{Y}'\mathbf{U}_X\Gamma_X\mathbf{V}_X'\mathbf{V}_X(f(\Gamma_X))^{-\frac{1}{2}}\mathbf{U}_F \\ &= \mathbf{V}_X(f(\Gamma_X))^{\frac{1}{2}}\mathbf{U}_F\Gamma_F\mathbf{V}_F'\mathbf{V}_F\Gamma_F\mathbf{U}_F'\mathbf{U}_F \\ &= \mathbf{V}_X(f(\Gamma_X))^{\frac{1}{2}}\mathbf{U}_F\Gamma_F^2 \end{aligned}$$

and

$$f(\mathbf{Q})\mathbf{T}\Lambda = \mathbf{V}\begin{bmatrix} f(\Gamma_X) & \mathbf{0} \\ \mathbf{0} & b_0\mathbf{I}_{p-r} \end{bmatrix}\mathbf{V}'\mathbf{V}_X(f(\Gamma_X))^{-\frac{1}{2}}\mathbf{U}_F\Gamma_F^2$$
$$= \mathbf{V}_X(f(\Gamma_X))^{\frac{1}{2}}\mathbf{U}_F\Gamma_F^2.$$

where $\mathbf{V} = [\mathbf{V}_X, \mathbf{V}_2]$ such that $\mathbf{V}_X'\mathbf{V}_2 = \mathbf{0}$. In addition, we have

$$\mathbf{T}'f(\mathbf{Q})\mathbf{T} = \mathbf{I}_r \quad \text{and} \quad \mathbf{T}'\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}\mathbf{T} = \Gamma_F^2.$$

References

S. Akaho. A kernel method for canonical correlation analysis. In *International Meeting of Psychometric Society*, 2001.

F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12:2385–2404, 2000.

Y.-Q. Cheng, Y.-M. Zhuang, and J.-Y. Yang. Optimal Fisher discriminant analysis using the rank decomposition. *Pattern Recognition*, 25(1):101–111, 1992.

M. Craven, D. Dopasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *The Fifteenth Conference on Artificial Intelligence*, 1998.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, New York, second edition, 2001.

J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1996.

T. Hastie, R. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89(428):1255–1270, 1994.

T. Hastie, A. Buja, and R. Tibshirani. Penalized discriminant analysis. *Annals of Statistics*, 23(1): 73–102, 1995.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York, 2001.

A. E. Hoerl and R. W. Kennard. Ridge regression. *Technometrics*, 12:56–82, 1970.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

P. Howland and H. Park. Generalizing discriminant analysis using the generalized singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):995–1006, 2004.

P. Howland, M. Jeon, and H. Park. Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 25(1):165–179, 2003.

J. Kittler and P. C. Young. A new approach to feature selection based on the Karhunen-Loève expansion. *Pattern Recognition*, 5:335–352, 1973.

K. C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):684–698, 2005.

H. Lütkepohl. *Handbook of Matrices*. John Wiley & Sons, New York, 1996.

K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, New York, 1979.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. Smola, and K. R. Müller. Invariant feature extraction and classification in kernel space. In *Advances in Neural Information Processing Systems 12*, volume 12, pages 526–532, 2000.

C. C. Paige and M. A. Saunders. Towards a generalized singular value decomposition. *SIAM Journal on Numerical Analysis*, 18(3):398–405, 1981.

C. H. Park and H. Park. Nonlinear discriminant analysis using kernel functions and the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 27(1):87–102, 2005a.

C. H. Park and H. Park. A relationship between linear discriminant analysis and the generalized minimum squared error solution. *SIAM Journal on Matrix Analysis and Applications*, 27(2): 474–492, 2005b.

K. Pelckmans, J. De Brabanter, J. A. K. Suykens, and B. De Moor. The differogram: Nonparametric noise variance estimation and its use for model. *Neurocomputing*, 69:100–122, 2005.

V. Roth and V. Steinhage. Nonlinear discriminant analysis using kernel functions. In *Advances in Neural Information Processing Systems 12*, volume 12, pages 568–574, 2000.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 1999.

J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

C. F. Van Loan. Generalizing the singular value decomposition. *SIAM Journal on Numerical Analysis*, 13(3):76–83, 1976.

T. Van Gestel, J. A. K. Suykens, J. De Brabanter, B. De Moor, and J. Vandewalle. Kernel canonical correlation analysis and least squares support vector machines. In *The International Conference on Artificial Neural Networks (ICANN)*, pages 381–386, 2001.

T. Van Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle. Bayesian framework for least-squares support vector machine classifiers, Gaussian processes, and kernel Fisher discriminant analysis. *Neural Computation*, 14:1115–1147, 2002.

A. R. Webb. *Statistical Pattern Recognition*. John Wiley & Sons, Hoboken, NJ, 2002.

J. Ye. Least squares linear discriminant analysis. In *The Twenty-Fourth International Conference on Machine Learning (ICML)*, 2007.

J. Ye, Q. Li, H. Xiong, H. Park, R. Janardan, and V. Kumar. An incremental dimension reduction algorithm via QR decomposition. In *ACM SIGKDD*, pages 364–373, 2004.

Z. Zhang and G. Dai. Optimal scoring for unsupervised learning. In *Advances in Neural Information Processing Systems 23*, volume 12, pages 2241–2249, 2009.

Z. Zhang and M. I. Jordan. Multiway spectral clustering: A margin-based perspective. *Statistical Science*, 3:383–403, 2008.

Journal of Machine Learning Research 11 (2010) 2229-2240 Submitted 11/09; Revised 5/10; Published 8/10

Erratum: SGDQN is Less Careful than Expected

Antoine Bordes ANTOINE.BORDES@LIP6.FR
LIP6 - Université Pierre et Marie Curie
4, Place Jussieu
75005 Paris, France

Léon Bottou LEON@BOTTOU.ORG
NEC Laboratories America, Inc.
4 Independence Way
Princeton, NJ 08540, USA

Patrick Gallinari PATRICK.GALLINARI@LIP6.FR
LIP6 - Université Pierre et Marie Curie
4, Place Jussieu
75005 Paris, France

Jonathan Chang JONCHANG@FACEBOOK.COM
S. Alex Smith ASMITH@FACEBOOK.COM
Facebook
1601 S. California Avenue
Palo Alto, CA 94304, USA

Editors: Soeren Sonnenburg, Vojtech Franc, Elad Yom-Tov and Michele Sebag

Abstract

The SGD-QN algorithm described in Bordes et al. (2009) contains a subtle flaw that prevents it from reaching its design goals. Yet the flawed SGD-QN algorithm has worked well enough to be a winner of the first Pascal Large Scale Learning Challenge (Sonnenburg et al., 2008). This document clarifies the situation, proposes a corrected algorithm, and evaluates its performance.

Keywords: stochastic gradient descent, support vector machine, conditional random fields

1. Introduction

Bordes et al. (2009) propose to improve the practical speed of stochastic gradient descent by efficiently estimating a diagonal matrix for rescaling the gradient estimates. The proposed algorithm, SGD-QN, works well enough to be a winner of the first Pascal Large Scale Learning Challenge (Sonnenburg et al., 2008). A couple months after the publication of the paper, Jonathan Chang and S. Alex Smith contacted Léon Bottou regarding some curious aspects of the algorithm mathematics (see Section 4.1). This initial observation was then traced back to a more subtle flaw that prevents the proposed algorithm to truly reach is design objectives.

We first explain the flaw and present experimental results describing its consequences. Then we present a corrected algorithm and evaluate its performance for training both linear Support Vector Machines (SVMs) and Conditional Random Fields (CRFs). Finally we draw updated conclusions.

2. Setup

Consider a binary classification problem with examples $(\mathbf{x}, y) \in \mathbb{R}^d \times \{-1,+1\}$. Given a set of examples $\{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n)\}$, we obtain a linear SVM classifier by minimizing the cost

$$\mathcal{P}_n(\mathbf{w}) \;=\; \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(y_i \mathbf{w}^\top \mathbf{x}_i) \;=\; \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + \ell(y_i \mathbf{w}^\top \mathbf{x}_i)\right).$$

Each iteration of the SGD-QN algorithm consists of drawing an independent random example $(\mathbf{x}_t, y_t)$ from the training set and computing an updated parameter vector

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t - \frac{1}{t+t_0}\mathbf{B}\,\mathbf{g}_t(\mathbf{w}_t) \quad \text{with} \quad \mathbf{g}_t(\mathbf{w}_t) = \lambda \mathbf{w}_t + \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t)\, y_t\, \mathbf{x}_t \tag{1}$$

where $\mathbf{B}$ is a diagonal scaling matrix estimated on-the-fly.

In the following, expectations and probabilities refer to the discrete distribution describing the training examples randomly picked from the finite training set at each iteration. Let $\mathcal{F}_t$ denote the examples $\{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_{t-1}, y_{t-1})\}$ picked before reaching the t-th iteration.

We would like to find $\mathbf{B}$ such that

$$\mathbf{w}_{t+1} - \mathbf{w}_t = \mathbf{B}\left(\mathcal{P}_n'(\mathbf{w}_{t+1}) - \mathcal{P}_n'(\mathbf{w}_t) + \xi_t\right), \tag{2}$$

with an error term ξ_t verifying $\mathbb{E}[\xi_t | \mathcal{F}_t] = 0$. Following Schraudolph et al. (2007), we replace the computationally expensive gradients $\mathcal{P}_n'$ by the cheap stochastic estimates $\mathbf{g}_\tau(\mathbf{w}_t)$ and $\mathbf{g}_\tau(\mathbf{w}_{t+1})$ computed on a same single example $(\mathbf{x}_\tau, y_\tau)$,

$$\mathbf{w}_{t+1} - \mathbf{w}_t = \mathbf{B}\left(\mathbf{g}_\tau(\mathbf{w}_{t+1}) - \mathbf{g}_\tau(\mathbf{w}_t) + \zeta_t + \xi_t\right), \tag{3}$$

where ζ_t represents the additional error term introduced by this substitution. Estimating B with (2) or (3) leads to the same solution if we make sure that $\mathbb{E}[\zeta_t | \mathcal{F}_t] = 0$ as well.

The SGD-QN algorithm updates the diagonal elements of matrix $\mathbf{B}$ on-the-fly on the basis of the term-by-term ratios of the observed differences $\mathbf{w}_{t+1} - \mathbf{w}_t$ and $\mathbf{g}_\tau(\mathbf{w}_{t+1}) - \mathbf{g}_\tau(\mathbf{w}_t)$. The obvious choices $\tau = t$ and $\tau = t+1$ only require one additional gradient evaluation because the parameter update formula (1) demands the computation of all the gradients $\mathbf{g}_t(\mathbf{w}_t)$ anyway.

3. The Flaw

Let us now evaluate

$$\mathbb{E}[\zeta_t | \mathcal{F}_t] = \mathbb{E}\left[\mathcal{P}_n'(\mathbf{w}_{t+1}) - \mathcal{P}_n'(\mathbf{w}_t) - \mathbf{g}_\tau(\mathbf{w}_{t+1}) + \mathbf{g}_\tau(\mathbf{w}_t) \middle| \mathcal{F}_t\right].$$

Let us first consider the case $\tau = t+1$. Since $\mathbf{g}_{t+1}(\mathbf{w}_{t+1})$ is a function of $(\mathbf{x}_{t+1}, y_{t+1}, \mathbf{x}_t, y_t, \mathcal{F}_t)$,

$$\begin{aligned}
\mathbb{E}[\mathbf{g}_\tau(\mathbf{w}_{t+1}) | \mathcal{F}_t] &= \int \mathbf{g}_{t+1}(\mathbf{w}_{t+1})\, dP(\mathbf{x}_{t+1}, y_{t+1}, \mathbf{x}_t, y_t \,|\, \mathcal{F}_t) \\
&= \int \left[\int \mathbf{g}_{t+1}(\mathbf{w}_{t+1})\, dP(\mathbf{x}_{t+1}, y_{t+1})\right] dP(\mathbf{x}_t, y_t \,|\, \mathcal{F}_t).
\end{aligned}$$

Since the variables $\mathbf{w}_{t+1}$ and $(\mathbf{x}_{t+1}, y_{t+1})$ are independent, the inner integral above is simply the average of $\mathbf{g}_{t+1}(\mathbf{w}_{t+1})$ for all possible $(\mathbf{x}_{t+1}, y_{t+1})$ picked from the training set. Therefore

$$\mathbb{E}\left[\mathbf{g}_\tau(\mathbf{w}_{t+1}) \middle| \mathcal{F}_t\right] \;=\; \int \mathcal{P}'_n(\mathbf{w}_{t+1})\, dP(\mathbf{x}_t, y_t \,|\, \mathcal{F}_t) \;=\; \mathbb{E}\left[\mathcal{P}'_n(\mathbf{w}_{t+1}) \middle| \mathcal{F}_t\right].$$

Using a similar derivation for $\mathbb{E}\left[\mathbf{g}_\tau(\mathbf{w}_t) \middle| \mathcal{F}_t\right]$ with $\tau = t+1$, we can easily establish that $\mathbb{E}\left[\zeta_t \middle| \mathcal{F}_t\right] = 0$. Therefore estimating $\mathbf{B}$ on the basis of (3) leads to the same solution as estimating $\mathbf{B}$ on the basis of (2), albeit with a higher noise level.

Such a derivation is impossible when $\tau = t$ because $(\mathbf{x}_t, y_t)$ and $\mathbf{w}_{t+1}$ are not independent. Therefore we cannot ensure that estimating $\mathbf{B}$ with (2) or (3) leads to the same result. Unfortunately, the SGD-QN paper (see Section 5.3 of Bordes et al., 2009) describes the algorithm with $\tau = t$.

4. The Consequences

In order to take maximal advantage of sparse data sets, the Flawed SGD-QN algorithm (see Figure 2 in the original paper) splits the stochastic parameter update (1) in two halves in order to schedule them separately. The first half involves only the gradient of the loss,

$$\mathbf{w} \leftarrow \mathbf{w} - (t+t_0)^{-1} \mathbf{B}\, \ell'(y_t\, \mathbf{w}^\top \mathbf{x}_t)\, y_t\, \mathbf{x}_t\,.$$

The second half involves only the gradient of the regularization term,

$$\mathbf{w} \leftarrow \begin{cases} \mathbf{w} & \text{most of the time,} \\ \mathbf{w} - \mathtt{skip}\, \lambda\, (t+t_0)^{-1} \mathbf{B}\, \mathbf{w} & \text{once every } \mathtt{skip} \text{ iterations.} \end{cases}$$

The Flawed SGD-QN algorithm measures the differences $\mathbf{w}_{t+1} - \mathbf{w}_t$ and $\mathbf{g}_t(\mathbf{w}_{t+1}) - \mathbf{g}_t(\mathbf{w}_t)$ during iterations for which the second half does nothing. Therefore, using notations $[\mathbf{x}]_i$ for the i-th coefficient of vector $\mathbf{x}$, and B_{ii} for the terms of the diagonal matrix $\mathbf{B}$, we always have

$$\frac{[\mathbf{g}_t(\mathbf{w}_{t+1}) - \mathbf{g}_t(\mathbf{w}_t)]_i}{[\mathbf{w}_{t+1} - \mathbf{w}_t]_i} \;=\; \lambda - \frac{\left(\ell'(y_t \mathbf{w}_{t+1}^\top \mathbf{x}_t) - \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t)\right) y_t\, [\mathbf{x}_t]_i}{B_{ii}\, (t+t_0)^{-1} \ell'(y_t \mathbf{w}^\top \mathbf{x}_t)\, y_t\, [\mathbf{x}_t]_i}\,.$$

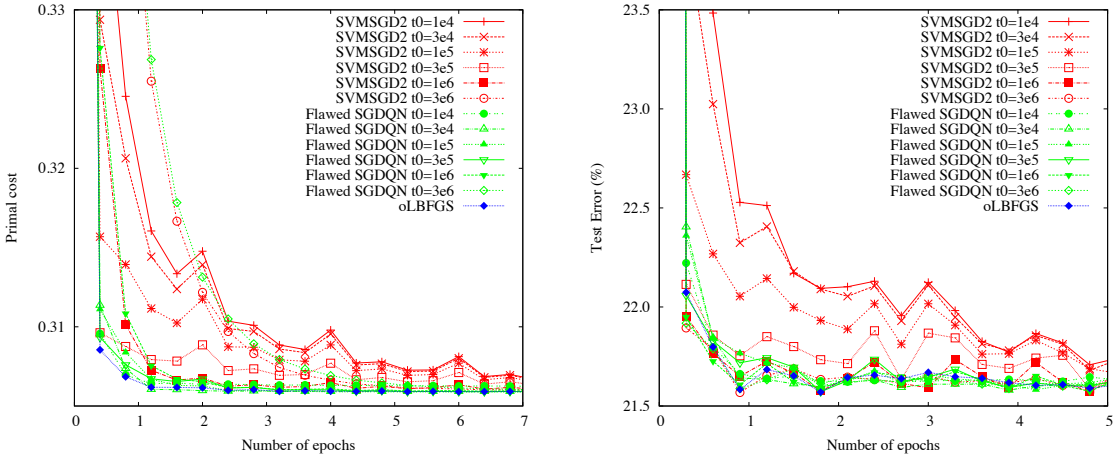
- When $[\mathbf{x}_t]_i$ is nonzero, we can simplify this expression as

$$\frac{[\mathbf{g}_t(\mathbf{w}_{t+1}) - \mathbf{g}_t(\mathbf{w}_t)]_i}{[\mathbf{w}_{t+1} - \mathbf{w}_t]_i} \;=\; \lambda - \frac{\ell'(y_t \mathbf{w}_{t+1}^\top \mathbf{x}_t) - \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t)}{B_{ii}\, (t+t_0)^{-1} \ell'(y_t \mathbf{w}_t^\top \mathbf{x}_t)}\,. \tag{4}$$

  This ratio is always greater than λ because of the loss function ℓ is convex. As explained in the original paper, the coefficients B_{ii} then remain smaller than λ^{-1}.

- When $[\mathbf{x}_t]_i$ is zero, the original paper uses a continuity argument to justify the equality

$$\frac{[\mathbf{g}_t(\mathbf{w}_{t+1}) - \mathbf{g}_t(\mathbf{w}_t)]_i}{[\mathbf{w}_{t+1} - \mathbf{w}_t]_i} \;=\; \lambda. \tag{5}$$



DELTA DATA SET

Figure 1: Plots of the training cost and test misclassification percentage versus the number of epochs for SVMSGD2 (red) and Flawed SGD-QN (green) for various values of t_0 on the dense Delta data set. The Flawed SGD-QN algorithm never outperforms the best SVMSGD2.

4.1 Impact on Dense Data Sets

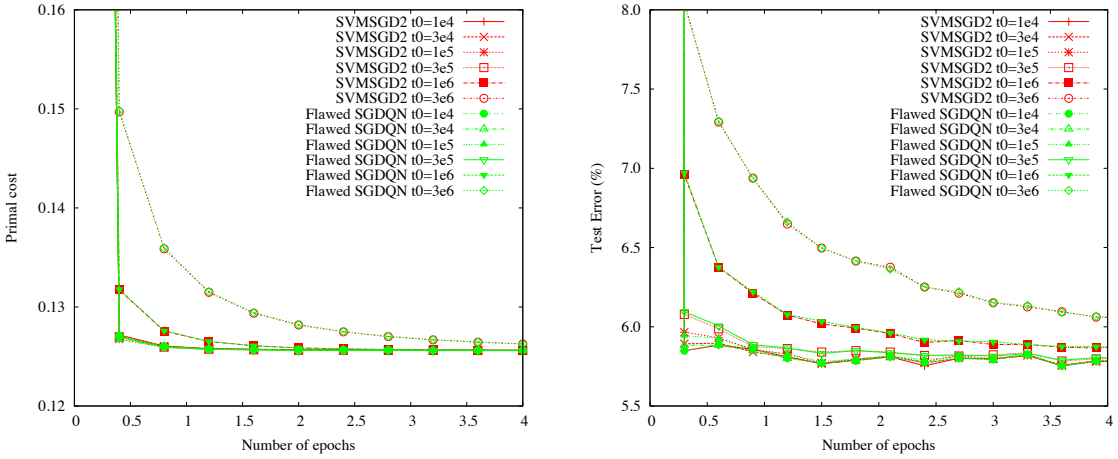
The coefficients $[\mathbf{x}_t]_i$ for dense data sets are rarely zero. Assume all the B_{ii} are equal before being updated. All the ratios (4) will then be equal. Therefore all the B_{ii} coefficients will be updated in exactly the same way and therefore remain equal. Since the B_{ii} coefficients are initially equal, they *remain equal all the time*, except maybe when encountering an occasional zero in the patterns $\mathbf{x}_t$. This observation led to the discovery of the flaw.

Since the scaling matrix reduces to a scalar gain, similar results could in principle be obtained using the ordinary stochastic gradient descent with a better gain schedule. This clearly defeats the purpose of the SGD-QN algorithm design.

Figure 1 compares the evolutions of the training cost and the test misclassification error for the SVMSGD2 and the Flawed SGD-QN algorithms for selected values of the parameter t_0 instead of the usual heuristic defaults. We observe that there is almost always a choice of t_0 in SVMSGD2 that performs as well as the best choice of t_0 for the Flawed SGD-QN. Both algorithms perform identically poorly for excessive values of t_0. On the other hand, when t_0 is too small, the performance of Flawed SGD-QN degrades much more gracefully than the performance of SVMSGD2. In some cases, Flawed SGD-QN can even slightly outperforms SVMSGD2 because, despite the flaw, it can still update its learning rate on the course of learning. This explains partially why we have consistently obtained better results with the flawed algorithm.

4.2 Impact on Sparse Data Sets

The situation is more complex in the case of sparse data sets because there is special case for updating the B_{ii} coefficients when dealing with zero coefficients (5). As a result, the Flawed SGD-QN algorithm gives higher values to the scaling coefficients B_{ii} when the i-th feature is more likely



RCV1 DATA SET

Figure 2: Plots of the training cost and test misclassification error versus the number of epochs for both SVMSGD2 (red) and Flawed SGD-QN (green) running on the RCV1 data set. Both algorithms reach optimal performance after seeing half the training set.

to be zero. Since this is a sensible scaling for such data sets, the Flawed SGD-QN algorithm works relatively well in the presence of sparse features.

Figure 2 compares the SVMSGD2 and Flawed SGD-QN algorithms for many choices for the t_0 parameter on the Reuters RCV1 data set. Unfortunately there is nothing to see there. Both algorithms reach optimal performance after processing only one half of the training set.

In order to find a more challenging sparse data set, we have adapted both the SVMSGD2 and the Flawed SGD-QN algorithms for the optimization of Conditional Random Fields (Lafferty et al., 2001). This is an interesting case where preconditioning is difficult because the features are generated on the fly on the basis of position-independent templates.

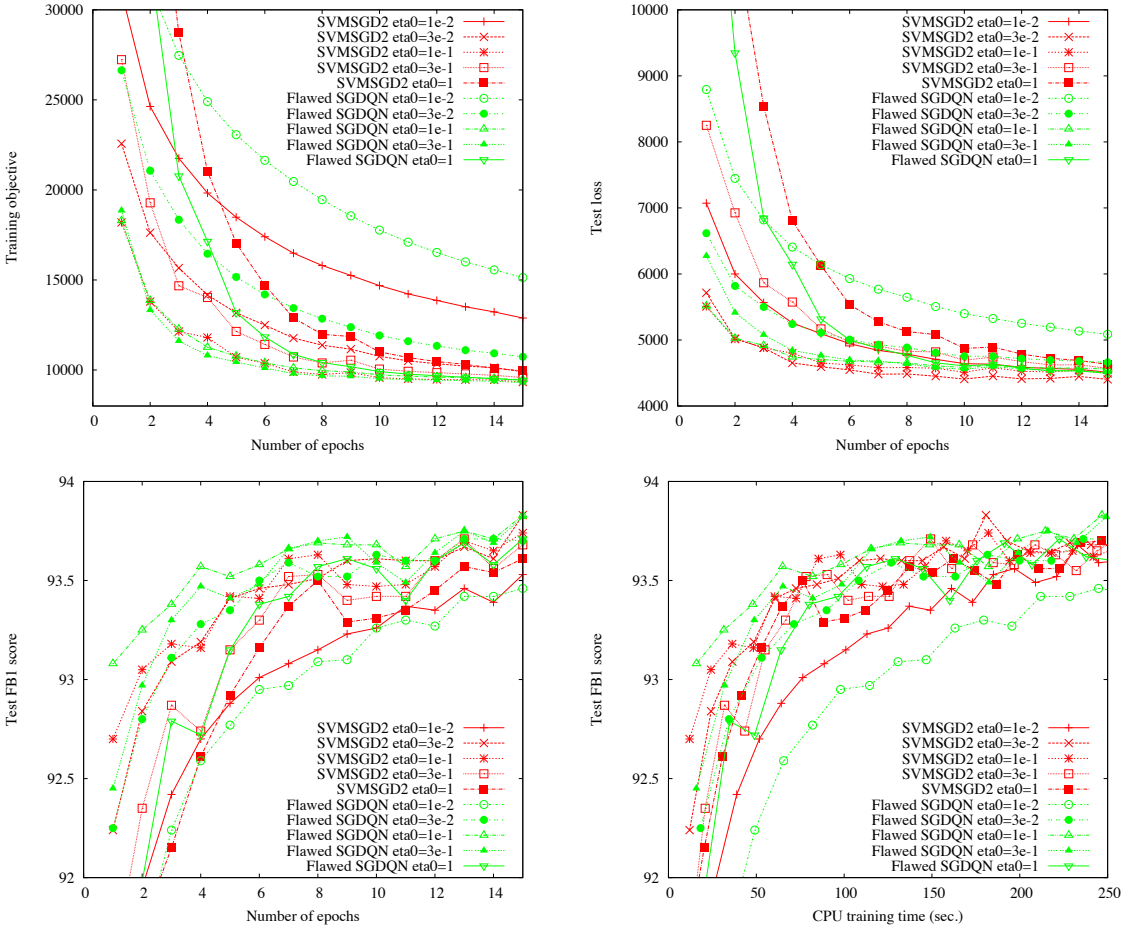
Figure 3 compares the algorithms on the CoNLL 2000 "chunking" task (Sang and Buchholz, 2000) using the template setup provided as an example with the CRF++ code (Kudo, 2007). The Flawed SGD-QN algorithm reaches the best test performance after less epochs than the SVMSGD2 algorithm, but this does not translate into a large improvement in terms of training time.

5. Correcting SGD-QN

At first glance, correcting SGD-QN simply involves computing the difference $\mathbf{g}_\tau(\mathbf{w}_{t+1}) - \mathbf{g}_\tau(\mathbf{w}_t)$ with $\tau = t+1$ instead of $\tau = t$. In fact, during the weeks preceding the Pascal challenge deadline, we tried both versions and found that picking $\tau = t+1$ performs significantly worse!

5.1 The Failure of the Straightforward Fix

When experimenting with the oLBFGS algorithm (Schraudolph et al., 2007), we observed and reported that setting the global learning gain was very difficult. We encounter the same difficulty when we modify the SGD-QN algorithm to use $\tau = t+1$.



CRF ON THE CONLL 2000 CHUNKING TASK

Figure 3: Plots of the training cost, test loss, and test F1 score for a CRF trained using both SVMSGD2 (red) and Flawed SGD-QN (green) for various initial rates $\eta_0 = \frac{1}{\lambda t_0}$.

In order to form an intuition about the learning rate, we must pay attention to its influence on the stochastic noise. Stochastic gradient descent with a constant learning rate generates a cloud of parameter estimates $\mathbf{w}_t$ covering a zone whose extent is defined by the learning rate and by the curvature of the cost function. When the rates decrease with an appropriate speed, this zone shrinks around the solution. Rewriting (1) term-by-term gives

$$[\mathbf{w}_{t+1}]_i \;=\; [\mathbf{w}_t]_i - \eta_{i,t}^{\text{flawQN}} \, [\mathbf{g}_t(\mathbf{w}_t)]_i \quad \text{with} \quad \eta_{i,t}^{\text{flawQN}} = \frac{B_{ii}}{t_0 + t} \,. \tag{6}$$

Since the algorithm periodically adapts B_{ii} on the basis of the observed differences $\mathbf{w}_{t+1} - \mathbf{w}_t$ and $\mathbf{g}_{t+1}(\mathbf{w}_{t+1}) - \mathbf{g}_{t+1}(\mathbf{w}_t)$, the sequence of learning rates $\eta_{i,t}^{\text{flawQN}}$ can occasionally increase. This is confirmed by the middle plot of Figure 5 which displays the evolution of the learning rates $\frac{B_{ii}}{t_0+t}$ for SGD-QN implementing this straightforward fix. Such fluctuations are the source of the difficulty.

Flawed SGD-QN	Corrected SGD-QN
Require: λ, t_0, T, skip	**Require:** λ, t_0, T, skip
$t \leftarrow 0$, $\mathbf{w} \leftarrow \mathbf{0}$, count $\leftarrow$ skip, $r \leftarrow 2$	$t \leftarrow 0$, $\mathbf{w} \leftarrow \mathbf{0}$, count $\leftarrow$ skip
$\mathbf{v} \leftarrow \mathbf{0}$, updateB $\leftarrow$ false, $\forall i \;\; B_{ii} \leftarrow (\lambda)^{-1}$	$\mathbf{v} \leftarrow \mathbf{0}$, updateB $\leftarrow$ false, $\forall i \;\; B_{ii} \leftarrow (\lambda t_0)^{-1}$
while $t \leq T$ **do**	**while** $t \leq T$ **do**
$\mathbf{w} \leftarrow \mathbf{w} - y_t \, \ell'(y_t \mathbf{w}^\top \mathbf{x}_t) \, (t+t_0)^{-1} \, \mathbf{B} \, \mathbf{x}_t$	
if updateB **then**	**if** updateB **then**
$\forall i \;\; r_i \leftarrow [\mathbf{g}_t(\mathbf{w}) - \mathbf{g}_t(\mathbf{v})]_i \,/\, [\mathbf{w} - \mathbf{v}]_i$	$\forall i \;\; r_i \leftarrow [\mathbf{g}_t(\mathbf{w}) - \mathbf{g}_t(\mathbf{v})]_i \,/\, [\mathbf{w} - \mathbf{v}]_i$
$\forall i \;\; r_i \leftarrow \min\{r_i, 100\lambda\}\}$	$\forall i \;\; r_i \leftarrow \max\{\lambda, \min\{100\lambda, r_i\}\}$
$\forall i \;\; B_{ii} \leftarrow B_{ii} + \frac{2}{r}(r_i^{-1} - B_{ii})$	$\forall i \;\; B_{ii} \leftarrow B_{ii}(1 + \text{skip} \, B_{ii} \, r_i)^{-1}$
updateB $\leftarrow$ false, $r \leftarrow r+1$	updateB $\leftarrow$ false
end if	**end if**
	$z \leftarrow y_t \mathbf{w}^\top \mathbf{x}_t$
count $\leftarrow$ count -1	count $\leftarrow$ count -1
if count ≤ 0 **then**	**if** count ≤ 0 **then**
count $\leftarrow$ skip, updateB $\leftarrow$ true	count $\leftarrow$ skip, updateB $\leftarrow$ true
$\mathbf{w} \leftarrow \mathbf{w} - \text{skip} \, \lambda \, (t+t_0)^{-1} \, \mathbf{B} \, \mathbf{w}$	$\mathbf{v} \leftarrow \mathbf{w}$
$\mathbf{v} \leftarrow \mathbf{w}$	$\mathbf{w} \leftarrow \mathbf{w} - \text{skip} \, \lambda \, \mathbf{B} \, \mathbf{w}$
end if	**end if**
	$\mathbf{w} \leftarrow \mathbf{w} - y_t \, \ell'(z) \, \mathbf{B} \, \mathbf{x}_t$
$t \leftarrow t+1$	$t \leftarrow t+1$
end while	**end while**
return $\mathbf{w}$	**return** $\mathbf{w}$

Figure 4: Pseudo-codes for the Flawed SGD-QN and Corrected SGD-QN algorithms. The main changes have been colored: each color stands for a particular change.

5.2 Managing the Speed of the Learning Rate Decrease

The schedule with which the learning rate decreases during training appears to be a key factor, so we propose to fix SGD-QN by using the second-order information to manage this diminution. Hence, we use learning rates of the form

$$\eta_{i,t}^{\text{corQN}} = \left(\lambda t_0 + \sum_{k=1}^{t-1} r_{i,k} \right)^{-1} \quad \text{where} \quad r_{i,t} = \frac{[\mathbf{g}_{t+1}(\mathbf{w}_{t+1}) - \mathbf{g}_{t+1}(\mathbf{w}_t)]_i}{[\mathbf{w}_{t+1} - \mathbf{w}_t]_i} \,. \tag{7}$$

When t becomes large, we recover an expression comparable to the original formulation (6), $\eta_{i,t}^{\text{corQN}} = \frac{\bar{r}_i^{-1}}{\lambda t_0 \bar{r}_i^{-1} + t} + \mathrm{o}\left(\frac{1}{t}\right)$, where $\bar{r}_i$ denotes the average value of the ratios $r_{i,t}$ and can be viewed as the coefficient of a diagonal matrix $\mathbf{R}$ such that $\mathbf{R}\left(\mathbf{w}_{t+1} - \mathbf{w}_t\right) = \mathbf{g}_\tau(\mathbf{w}_{t+1}) - \mathbf{g}_\tau(\mathbf{w}_t) + \zeta_t + \xi_t$.

It is also interesting to compare the formula $\eta_{i,t}^{\text{corQN}}$ with the first order version $\eta_{i,t}^{\text{SGD}} = \frac{1}{\lambda t_0 + \sum_{k=1}^{t} \lambda}$ which decreases the learning rate after each iteration by adding λ to the denominator. Instead of adding a lower bound of the curvature, the proposed learning rate formula adds a stochastic estimate of the curvature.

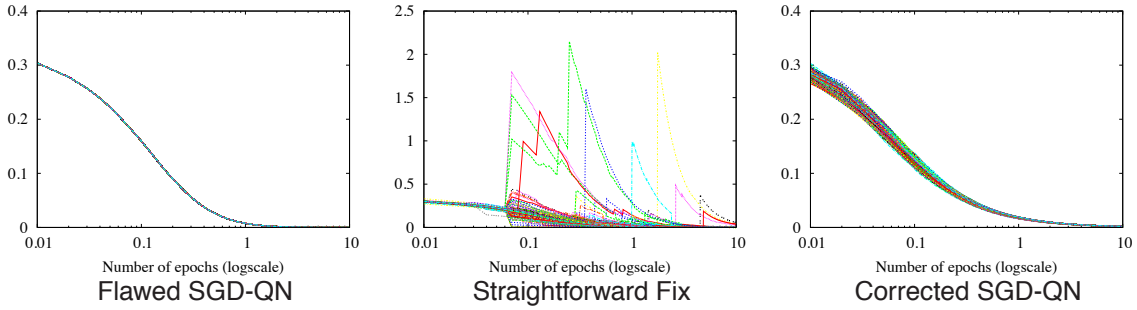


Figure 5: Plots of the learning rates corresponding to each feature on the course of learning on the Delta data set. All rates are equal for Flawed SGD-QN (left plot). As explained in Section 5.1, implementing the straightforward fix (middle plot) causes rates to alternatively increase or decrease very fast. The Corrected SGD-QN (right plot) proposes learning rates nicely decreasing at different speeds for each feature.

Interestingly, Equation (7) leads to a convenient recursive formula

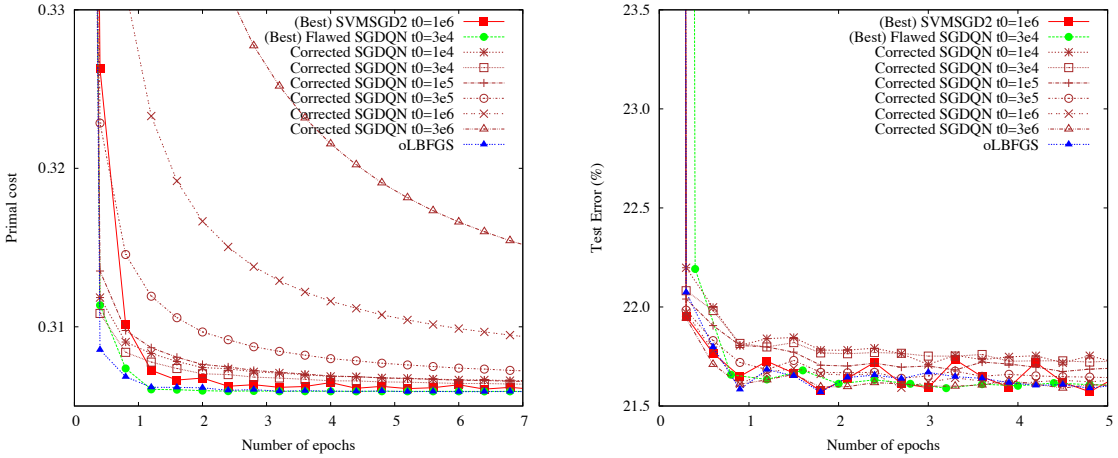
$$\eta_{i,t}^{\text{corQN}} = \left(\frac{1}{\eta_{i,t-1}^{\text{corQN}}} + r_{i,t-1} \right)^{-1} = \frac{\eta_{i,t-1}^{\text{corQN}}}{1 + r_{i,t-1}\, \eta_{i,t-1}^{\text{corQN}}} . \tag{8}$$

Figure 4 describes the Corrected SGD-QN algorithm and compares it with a slightly reorganized version of the Flawed SGD-QN algorithm. The diagonal matrix $\mathbf{B}$ is used to store the gains (7). The algorithm schedules a gain update (line 14) whenever it performs a regularization update (line 16). During the next iteration, the algorithm computes $r_{i,t-1}$ (line 6) and implements the learning rate update (8) with an additional multiplicative factor (line 8) because this only happens every `skip` iterations. The effect on the learning rates of using Corrected SGD-QN instead of Flawed SGD-QN is illustrated by Figure 5 if we compare the left and the right plots.

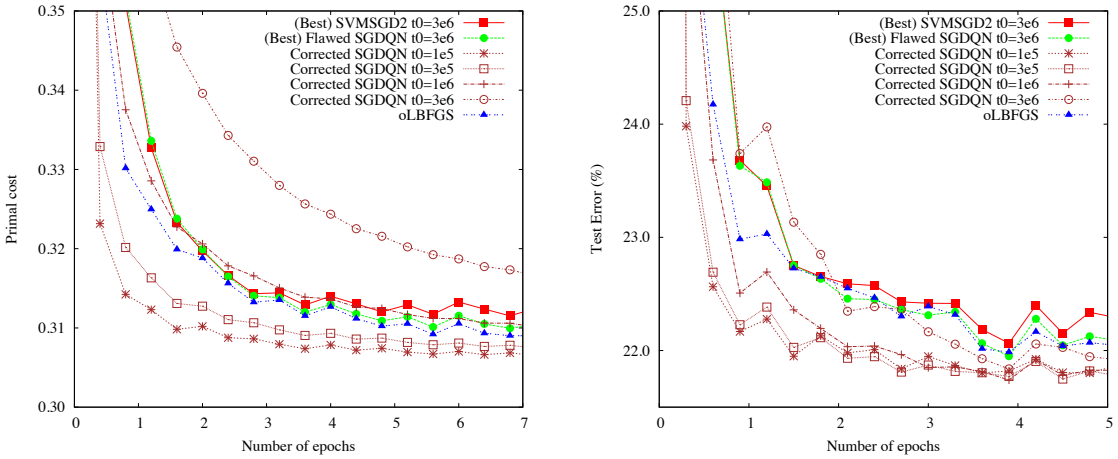
5.3 Performances on Dense Data Sets

Figure 6 (top row) compares the performances of Corrected SGD-QN with the best results of Flawed SGD-QN and SVMSGD2 on the Delta data set. We must recognize that the improvement is minimal. Before running the SGD algorithms, we always precondition the dense data sets by centering all the features, normalizing their variances, and rescaling every example to ensure that $\|\mathbf{x}_k\| = 1$. This operation in fact steals all the improvements SGD-QN can bring. With its adaptive learning rates, the Corrected SGD-QN does not perform worse than the first order SVMSGD2 algorithm. Yet, implementing a strategy involving a single learning rate for all the features appears already very rewarding and, for such cases, the Flawed SGD-QN algorithm is a strong choice because of its capacity to adapt its learning rate.

Corrected SGD-QN should be more efficient for ill-conditioned data. To illustrate this assertion, we created a "deconditioned" version of Delta by applying the usual normalization procedures and then multiplying every tenth feature by twelve. Figure 6 (bottom row) compares the performances of SVMSGD2, Flawed SGD-QN and Corrected SGD-QN on this deconditioned data. The Flawed



DELTA DATA SET (NORMALIZED)



DELTA DATA SET (DECONDITIONED)

Figure 6: Plots of the training cost and test misclassification error versus the number of epochs for SVMSGD2 (red) and Flawed SGD-QN (green) with their optimal t_0 parameter, and Corrected SGD-QN (brown) running on the Delta data set. Both *normalized* (top) and *deconditioned* (bottom) cases are considered; see the text for details. All methods can perform roughly identically well on normalized examples but only the Corrected SGD-QN algorithm is able to handle ill-conditioned data.

SGD-QN algorithm clearly suffers from the deconditioning operation because it can not assign a different learning rate per feature. The Corrected SGD-QN works much better. We also verified that the estimated learning rates replicate the deconditioning pattern.

In conclusion, on *dense data sets*, the Corrected SGD-QN bring little improvement over those associated with a *good preconditioning technique*. Preconditioning was probably the main reason of the good SGD-QN results on dense data sets in the Pascal Large Scale Challenge. This does not

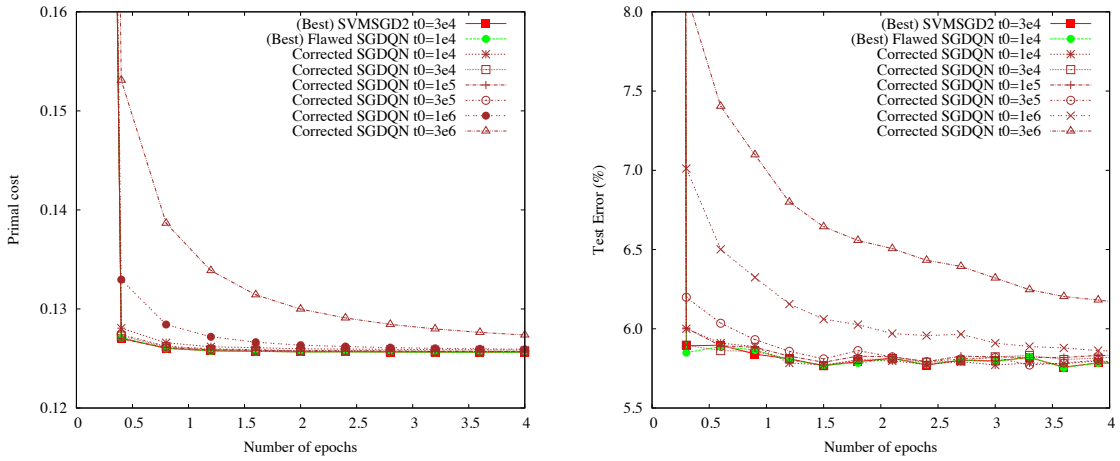


Figure 7: Plots of the training cost and test error versus the number of epochs for SVMSGD2 (red) and Flawed SGD-QN (green) with their optimal t_0 parameter, and Corrected SGD-QN (brown) running on the RCV1 data set. All algorithms quickly reach optimal performance.

mean that SGD algorithms cannot be improved. Xu (2010) reports impressive results on Linear SVMs using a well sorted Averaged SGD algorithm (Polyak and Juditsky, 1992).

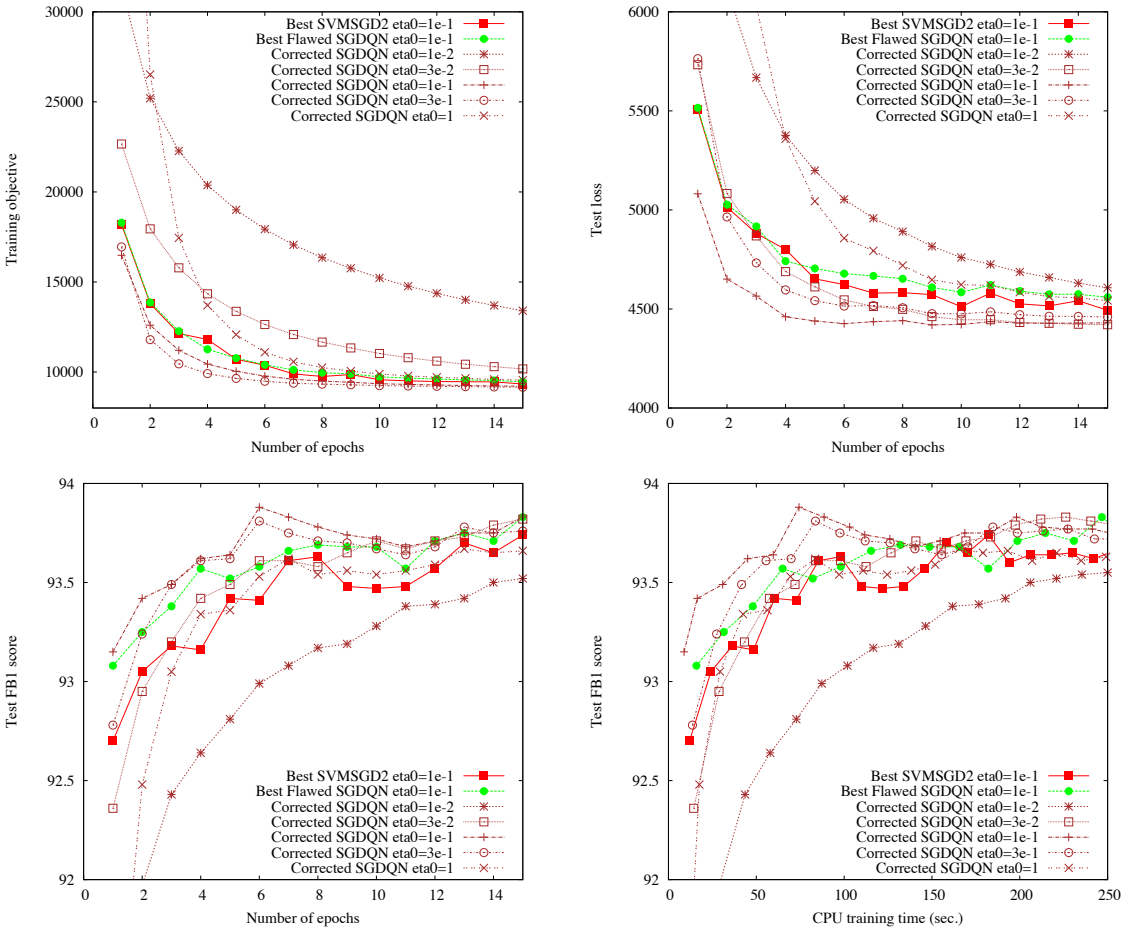
5.4 Performances on Sparse Data Sets

Preconditioning sparse data sets is much more difficult because it is impossible to center sparse features and keep them sparse. In addition, normalizing the variance of very rare features generates a small number of coefficients with high values. This fat tail distribution usually has very negative impact on the test performance. Figure 7 compares the SVMSGD2, Flawed SGD-QN and Corrected SGD-QN algorithms on the Reuters RCV1 data set, but, as we explained for Figure 2, this task is too easy to draw any conclusions.

Figure 8 then compares the adaptations of SVMSGD2, Flawed SGD-QN (with their best parameters) and Corrected SGD-QN for Conditional Random Fields on the CoNLL 2000 "chunking" task with the setup described in Section 4.2. The Corrected SGD-QN algorithm achieves its optimal test performance after only 75 seconds while SVMSGD2 and Flawed SGD-QN need around twice this time. For comparison, the CRF++ LBFGS optimizer needs 4300 seconds on a slightly faster machine.

6. Conclusion

Despite its flaw, the original SGD-QN algorithm works well enough to be a winner of the first PASCAL Large Scale Learning Challenge (Sonnenburg et al., 2008) because it benefits from our careful preconditioning and handles sparse examples efficiently. However, as explained in this document, this original version often does not achieve the full benefits of a diagonal scaling approach.



CRF ON THE CONLL 2000 CHUNKING TASK

Figure 8: Plots of the training cost, test loss, and test F1 score for a CRF trained using the best setups of SVMSGD2 (red) and Flawed SGD-QN (green), and Corrected SGD-QN for various initial rates $\eta_0 = \frac{1}{\lambda t_0}$ (brown). Corrected SGD-QN learns significantly faster.

This paper proposes a correction. Unlike the original SGD-QN algorithm, the Corrected SGD-QN algorithm discovers sensible diagonal scaling coefficients. However, experiments on dense data sets of intermediate dimensionality show that similar speed improvements can be achieved by simple preconditioning techniques such as normalizing the means and the variances of each feature and normalizing the length of each example. On the other hand, normalization is not always an attractive strategy. The Corrected SGD-QN algorithm then becomes interesting because it can adapt automatically to skewed feature distributions (see Section 5.3) or very sparse data (see Section 5.4.)

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. Part of this work was funded by the EU Network of Excellence PASCAL2 and by the French DGA.

References

A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, July 2009.

T. Kudo. CRF++: Yet another CRF toolkit, 2007. `http://crfpp.sourceforge.net`.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289, Williams College, Williamstown, 2001. Morgan Kaufmann.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, 1992.

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132. Lisbon, Portugal, 2000.

N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. In *Proc. 11th Intl. Conf. on Artificial Intelligence and Statistics (AIstats)*, pages 433–440. Soc. for Artificial Intelligence and Statistics, 2007.

S. Sonnenburg, V. Franc, E. Yom-Tov, and M. Sebag. Pascal large scale learning challenge. ICML 2008 Workshop, 2008. `http://largescale.first.fraunhofer.de`.

W. Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. Submitted to JMLR, 2010.

Journal of Machine Learning Research 11 (2010) 2241-2259 Submitted 3/10; Revised 8/10; Published 8/10

Restricted Eigenvalue Properties for Correlated Gaussian Designs

Garvesh Raskutti GARVESHR@STAT.BERKELEY.EDU
Martin J. Wainwright* WAINWRIG@STAT.BERKELEY.EDU
Bin Yu* BINYU@STAT.BERKELEY.EDU
Department of Statistics
University of California
Berkeley, CA 94720-1776, USA

Editor: John Lafferty

Abstract

Methods based on ℓ_1-relaxation, such as basis pursuit and the Lasso, are very popular for sparse regression in high dimensions. The conditions for success of these methods are now well-understood: (1) exact recovery in the noiseless setting is possible if and only if the design matrix X satisfies the restricted nullspace property, and (2) the squared ℓ_2-error of a Lasso estimate decays at the minimax optimal rate $\frac{k \log p}{n}$, where k is the sparsity of the p-dimensional regression problem with additive Gaussian noise, whenever the design satisfies a restricted eigenvalue condition. The key issue is thus to determine when the design matrix X satisfies these desirable properties. Thus far, there have been numerous results showing that the restricted isometry property, which implies both the restricted nullspace and eigenvalue conditions, is satisfied when all entries of X are independent and identically distributed (i.i.d.), or the rows are unitary. This paper proves directly that the restricted nullspace and eigenvalue conditions hold with high probability for quite general classes of Gaussian matrices for which the predictors may be highly dependent, and hence restricted isometry conditions can be violated with high probability. In this way, our results extend the attractive theoretical guarantees on ℓ_1-relaxations to a much broader class of problems than the case of completely independent or unitary designs.

Keywords: Lasso, basis pursuit, random matrix theory, Gaussian comparison inequality, concentration of measure

1. Introduction

Many fields in modern science and engineering—among them computational biology, astrophysics, medical imaging, natural language processing, and remote sensing—involve collecting data sets in which the dimension of the data p exceeds the sample size n. Problems of statistical inference in this high-dimensional setting have attracted a great deal of attention in recent years. One concrete instance of a high-dimensional inference problem concerns the standard linear regression model, in which the goal is to estimate a vector $\beta^* \in \mathbb{R}^p$ that connects a real-valued response y to a vector of covariates $X = (X_1, \ldots, X_p)$. In the setting $p \gg n$, the classical linear regression model is unidentifiable, so that it is not meaningful to estimate the parameter vector $\beta^* \in \mathbb{R}^p$. However, many high-dimensional regression problems exhibit special structure that can lead to an identifiable model. In particular, sparsity in the regression vector β^* is an archetypal example of such struc-

*. Also in the Department of Electrical Engineering & Computer Science.

ture, and there is now a substantial and rapidly growing body of work on high-dimensional linear regression with sparsity constraints.

Using the ℓ_1-norm to enforce sparsity has been very successful, as evidenced by the widespread use of methods such as basis pursuit (Chen et al., 1998), the Lasso (Tibshirani, 1996) and the Dantzig selector (Candes and Tao, 2007). There is now a well-developed theory on what conditions are required on the design matrix $X \in \mathbb{R}^{n \times p}$ for such ℓ_1-based relaxations to reliably estimate β^*. In the case of noiseless observation models, it is known that imposing a *restricted nullspace property* on the design matrix $X \in \mathbb{R}^{n \times p}$ is both necessary and sufficient for the basis pursuit linear program to recover β^* exactly. The nullspace property and its link to the basis pursuit linear program has been discussed in various papers (Cohen et al., 2009; Donoho and Huo, 2001; Feuer and Nemirovski, 2003). In the case of noisy observations, exact recovery of β^* is no longer possible, and one goal is to obtain an estimate $\widehat{\beta}$ such that the ℓ_2-error $\|\widehat{\beta} - \beta^*\|_2$ is well-controlled. To this end, various sufficient conditions for the success of ℓ_1-relaxations have been proposed, including restricted eigenvalue conditions (Bickel et al., 2009; Meinshausen and Yu, 2009) and the restricted Riesz property (Zhang and Huang, 2008). Of the conditions mentioned, one of weakest known sufficient conditions for bounding ℓ_2-error are the restricted eigenvalue (RE) conditions due to Bickel et al. (2009) and van de Geer (2007). In this paper, we consider a restricted eigenvalue condition that is essentially equivalent to the RE condition in Bickel et al. (2009). As shown by Raskutti et al. (2009), a related restriction is actually necessary for obtaining good control on the ℓ_2-error in the minimax setting.

Thus, in the setting of linear regression with random design, the interesting question is the following: for what ensembles of design matrices do the restricted nullspace and eigenvalue conditions hold with high probability? To date, the main routes to establishing these properties have been via either incoherence conditions (Donoho and Huo, 2001; Feuer and Nemirovski, 2003) or via the restricted isometry property (Candes and Tao, 2005), both of which are sufficient but not necessary conditions (Cohen et al., 2009; van de Geer and Buhlmann, 2009). The restricted isometry property (RIP) holds with high probability for various classes of random matrices with i.i.d. entries, including sub-Gaussian matrices (Mendelson et al., 2008) with sample size $n = \Omega(k \log(p/k))$, and for i.i.d. subexponential random matrices (Adamczak et al., 2009) provided that $n = \Omega(k \log^2(p/k))$. It has also been demonstrated that RIP is satisfied for matrices from unitary ensembles (e.g., Guédon et al., 2007, 2008; Romberg, 2009; Rudelson and Vershynin, 2008), for which the rows are generated based on independent draws from a set of uncorrelated basis functions.

Design matrices based on i.i.d. or unitary ensembles are well-suited to the task of compressed sensing (Candes and Tao, 2005; Donoho, 2006), where the matrix X can be chosen by the user. However, in most of machine learning and statistics, the design matrix is not under control of the statistician, but rather is specified by nature. As a concrete example, suppose that we are fitting a linear regression model to predict heart disease on the basis of a set of p covariates (e.g., diet, exercise, smoking). In this setting, it is not reasonable to assume that the different covariates are i.i.d. or unitary—for instance, one would expect a strong positive correlation between amount of exercise and healthiness of diet. Nonetheless, at least in practice, ℓ_1-methods still work very well in settings where the covariates are correlated and non-unitary, but currently lacking is the corresponding theory that guarantees the performance of ℓ_1-relaxations for dependent designs.

The main contribution of this paper is a direct proof that both the restricted nullspace and eigenvalue conditions hold with high probability for a broad class of dependent Gaussian design matrices. In conjunction with known results on ℓ_1-relaxation, our main result implies as corollaries that the

basis pursuit algorithm reliably recovers β^* exactly in the noiseless setting, and that in the case of observations contaminated by Gaussian noise, the Lasso and Dantzig selectors produces a solution $\widehat{\beta}$ such that $\|\widehat{\beta} - \beta^*\|_2 = O(\sqrt{\frac{k \log p}{n}})$. Our theory requires that the sample size n scale as $n = \Omega(k \log p)$, where k is the sparsity index of the regression vector β^* and p is its dimensions. For sub-linear sparsity ($k/p \to 0$), this scaling matches known optimal rates in a minimax sense for the sparse regression problem (Raskutti et al., 2009), and hence cannot be improved upon by any algorithm. The class of matrices covered by our result allows for correlation among different covariates, and hence covers many matrices for which restricted isometry or incoherence conditions fail to hold but the restricted eigenvalue condition holds. Interestingly, one can even sample the rows of the design matrix X from a multivariate Gaussian with a degenerate covariance matrix Σ, and nonetheless, our results still guarantee that the restricted nullspace and eigenvalue conditions will hold with high probability (see Section 3.3). Consequently, our results extend theoretical guarantees on ℓ_1-relaxations with optimal rates of convergence to a much broader class of random designs.

The remainder of this paper is organized as follows. We begin in Section 2 with background on sparse linear models, the basis pursuit and Lasso ℓ_1-relaxations, and sufficient conditions for their success. In Section 3, we state our main result, discuss its consequences for ℓ_1-relaxations, and illustrate it with some examples. Section 4 contains the proof of our main result, which exploits Gaussian comparison inequalities and concentration of measure for Lipschitz functions.

2. Background

We begin with background on sparse linear models and sufficient conditions for the success of ℓ_1-relaxations.

2.1 High-dimensional Sparse Models and ℓ_1-relaxation

In the classical linear model, a scalar output $y_i \in \mathbb{R}$ is linked to a p-dimensional vector $X_i \in \mathbb{R}^p$ of covariates via the relation $y_i = X_i^T \beta^* + w_i$, where w_i is a scalar observation noise. If we make a set of n such observations, then they can be written in the matrix-vector form

$$y \quad = \quad X\beta^* + w, \tag{1}$$

where $y \in \mathbb{R}^n$ is the vector of outputs, the matrix $X \in \mathbb{R}^{n \times p}$ is the set of covariates (in which row $X_i \in \mathbb{R}^p$ represents the covariates for i^{th} observation), and $w \in \mathbb{R}^n$ is a noise vector where $w \sim \mathcal{N}(0, \sigma^2 I_{n \times n})$. Given the pair (y, X), the goal is to estimate the unknown regression vector $\beta^* \in \mathbb{R}^p$.

In many applications, the linear regression model is high-dimensional in nature, meaning that the number of observations n may be substantially smaller than the number of covariates p. In this $p \gg n$ regime, it is easy to see that without further constraints on β^*, the statistical model (1) is not identifiable, since (even when $w = 0$), there are many vectors β^* that are consistent[1] with the observations y and X. This identifiability concern may be eliminated by imposing some type of sparsity assumption on the regression vector $\beta^* \in \mathbb{R}^p$. The simplest assumption is that of *exact sparsity*: in particular, we say that $\beta^* \in \mathbb{R}^p$ is s-sparse if its support set

$$S(\beta^*) := \{ j \in \{1, \ldots, p\} \mid \beta_j^* \neq 0 \}$$

1. Indeed, any vector β^* in the nullspace of X, which has dimension at least $p - n$, leads to $y = 0$ when $w = 0$.

has cardinality at most s.

Disregarding computational cost, the most direct approach to estimating an s-sparse β^* in the linear regression model would be solving a quadratic optimization problem with an ℓ_0-constraint, say

$$\widehat{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \qquad \text{such that } \|\beta\|_0 \leq s,$$

where $\|\beta\|_0$ simply counts the number of non-zero entries in β. Of course, this problem is non-convex and combinatorial in nature, since it involves searching over all $\binom{p}{s}$ subsets of size s. A natural relaxation is to replace the non-convex ℓ_0 constraint with the ℓ_1-norm, which leads to the *constrained form of the Lasso* (Chen et al., 1998; Tibshirani, 1996), given by

$$\widehat{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2 \qquad \text{such that } \|\beta\|_1 \leq R,$$

where R is a radius to be chosen by the user. Equivalently, by Lagrangian duality, this program can also be written in the penalized form

$$\widehat{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\},$$

where $\lambda > 0$ is a regularization parameter. In the case of noiseless observations, obtained by setting $w = 0$ in the observation model (1), a closely related convex program is the *basis pursuit linear program* (Chen et al., 1998), given by

$$\widehat{\beta} \in \arg\min_{\widehat{\beta} \in \mathbb{R}^p} \|\beta\|_1 \qquad \text{such that } X\beta = y. \tag{2}$$

Chen et al. (1998) also study the constrained Lasso (2.1), which they refer to as relaxed basis pursuit. Another closely related estimator based on ℓ_1-relaxation is the Dantzig selector (Candes and Tao, 2007).

2.2 Sufficient Conditions for Success

The high-dimensional linear model under the exact sparsity constraint has been extensively analyzed. Accordingly, as we discuss here, there is a good understanding of the necessary and sufficient conditions for the success of ℓ_1-based relaxations such as basis pursuit and the Lasso.

2.2.1 RESTRICTED NULLSPACE IN NOISELESS SETTING

In the noiseless setting ($w = 0$), it is known that the basis pursuit linear program (LP) (2) recovers β^* exactly if and only if the design matrix X satisfies a restricted nullspace condition. In particular, for a given subset $S \subset \{1, \ldots, p\}$ and constant $\alpha \geq 1$, let us define the set

$$\mathcal{C}(S; \alpha) := \left\{ \theta \in \mathbb{R}^p \mid \|\theta_{S^c}\|_1 \leq \alpha \|\theta_S\|_1 \right\}.$$

For a given sparsity index $k \leq p$, we say that the matrix X satisfies the *restricted nullspace (RN) condition* of order k if $\text{null}(X) \cap \mathcal{C}(S; 1) = \{0\}$ for all subsets S of cardinality k. Although this definition appeared in earlier work (Donoho and Huo, 2001; Feuer and Nemirovski, 2003), the terminology of restricted nullspace is due to Cohen et al. (2009).

This restricted nullspace property is important, because the basis pursuit LP recovers any vector k-sparse vector β^* exactly if and only if X satisfies the restricted nullspace property of order k. See the papers (Cohen et al., 2009; Donoho and Huo, 2001; Elad and Bruckstein, 2002; Feuer and Nemirovski, 2003) for more discussion of restricted nullspaces and equivalence to exact recovery of basis pursuit.

2.2.2 RESTRICTED EIGENVALUE CONDITION FOR ℓ_2 ERROR

In the noisy setting, it is impossible to recover β^* exactly, and a more natural criterion is to bound the ℓ_2-error between β^* and an estimate $\widehat{\beta}$. Various conditions have been used to analyze the ℓ_2-norm convergence rate of ℓ_1-based methods, including the restricted isometry property (Candes and Tao, 2007), various types of restricted eigenvalue conditions (van de Geer, 2007; Bickel et al., 2009; Meinshausen and Yu, 2009), and a partial Riesz condition (Zhang and Huang, 2008). Of all these conditions, the least restrictive are the restricted eigenvalue conditions due to Bickel et al. (2009) and van de Geer (2007). As shown by Bickel et al. (2009), their restricted eigenvalue (RE) condition is less severe than both the RIP condition (Candes and Tao, 2007) and an earlier set of restricted eigenvalue conditions due to Meinshausen and Yu (2009). All of these conditions involve lower bounds on $\|X\theta\|_2$ that hold uniformly over the previously defined set $\mathcal{C}(S;\alpha)$,

Here we state a condition that is essentially equivalent to the restricted eigenvalue condition due to Bickel et al. (2009). In particular, we say that the $p \times p$ sample covariance matrix $X^T X/n$ satisfies the *restricted eigenvalue (RE) condition* over S with parameters $(\alpha,\gamma) \in [1,\infty) \times (0,\infty)$ if

$$\frac{1}{n}\theta^T X^T X\theta \;=\; \frac{1}{n}\|X\theta\|_2^2 \;\geq\; \gamma^2\,\|\theta\|_2^2 \qquad \text{for all } \theta \in \mathcal{C}(S;\alpha).$$

If this condition holds uniformly for all subsets S with cardinality k, we say that $X^T X/n$ *satisfies a restricted eigenvalue condition of order k with parameters* (α,γ). On occasion, we will also say that a deterministic $p \times p$ covariance matrix Σ satisfies an RE condition, by which we mean that $\|\Sigma^{1/2}\theta\|_2 \geq \gamma\|\theta\|_2$ for all $\theta \in \mathcal{C}(S;\alpha)$. It is straightforward to show that the RE condition for some α implies the restricted nullspace condition for the same α, so that the RE condition is slightly stronger than the RN property.

Again, the RE condition is important because it yields guarantees on the ℓ_2-error of any Lasso estimate $\widehat{\beta}$. For instance, if X satisfies the RE condition of order k with parameters $\alpha \geq 3$ and $\gamma > 0$, then it can be shown that (with appropriate choice of the regularization parameter) any Lasso estimate $\widehat{\beta}$ satisfies the error bound $\|\widehat{\beta} - \beta^*\|_2 = O(\sqrt{\frac{k\log p}{n}})$ with high probability over the Gaussian noise vector w. A similar result holds for the Dantzig selector provided the RE condition is satisfied for $\alpha \geq 1$. Bounds with this scaling have appeared in various papers on sparse linear models (Bunea et al., 2007; Bickel et al., 2009; Candes and Tao, 2007; Meinshausen and Yu, 2009; van de Geer, 2007; van de Geer and Buhlmann, 2009). Moreover, this ℓ_2-convergence rate is known to be minimax optimal (Raskutti et al., 2009) in the regime $k/p \to 0$.

3. Main Result and Its Consequences

Thus, in order to provide performance guarantees (either exact recovery or ℓ_2-error bounds) for ℓ_1-relaxations applied to sparse linear models, it is sufficient to show that the RE or RN conditions hold. Given that our interest is in providing sufficient conditions for these properties, the remainder

of the paper focuses on providing conditions for the RE condition to hold for random designs, which implies that the RN condition is satisfied.

3.1 Statement of Main Result

Our main result guarantees that the restricted eigenvalue (and hence restricted nullspace) conditions hold for a broad class of random Gaussian designs. In particular, we consider the linear model $y_i = X_i^T \beta^* + w_i$, in which each row $X_i \sim \mathcal{N}(0, \Sigma)$. We define $\rho^2(\Sigma) = \max_{j=1,\ldots,p} \Sigma_{jj}$ to be the maximal variance, and let $\Sigma^{1/2}$ denote the square root of Σ.

Theorem 1 *For any Gaussian random design $X \in \mathbb{R}^{n \times p}$ with i.i.d. $\mathcal{N}(0, \Sigma)$ rows, there are universal positive constants c, c' such that*

$$\frac{\|Xv\|_2}{\sqrt{n}} \;\geq\; \frac{1}{4}\|\Sigma^{1/2} v\|_2 - 9\,\rho(\Sigma)\,\sqrt{\frac{\log p}{n}}\,\|v\|_1 \qquad \text{for all } v \in \mathbb{R}^p, \tag{3}$$

with probability at least $1 - c' \exp(-cn)$.

The proof of this claim is given later in Section 4. Note that we have not tried to obtain sharpest possible leading constants (i.e., the factors of $1/4$ and 9 can easily be improved).

In intuitive terms, Theorem 1 provides some insight into the eigenstructure of the sample covariance matrix $\widehat{\Sigma} = X^T X/n$. One implication of the lower bound (3) is that the nullspace of X cannot contain any vectors that are "overly" sparse. In particular, for any vector $v \in \mathbb{R}^p$ such that $\|v\|_1 / \|\Sigma^{1/2} v\|_2 = o(\sqrt{\frac{n}{\log p}})$, the right-hand side of the lower bound (3) will be strictly positive, showing that v cannot belong to the nullspace of X. In the following corollary, we formalize this intuition by showing how Theorem 1 guarantees that whenever the population covariance Σ satisfies the RE condition of order k, then the sample covariance $\widehat{\Sigma} = X^T X/n$ satisfies the same property as long as the sample size is sufficiently large.

Corollary 1 (Restricted eigenvalue property) *Suppose that Σ satisfies the RE condition of order k with parameters (α, γ). Then for universal positive constants c, c', c'', if the sample size satisfies*

$$n \;>\; c'' \,\frac{\rho^2(\Sigma)\,(1+\alpha)^2}{\gamma^2}\, k \log p, \tag{4}$$

then the matrix $\widehat{\Sigma} = X^T X/n$ satisfies the RE condition with parameters $(\alpha, \frac{\gamma}{8})$ with probability at least $1 - c' \exp(-cn)$.

Proof Let S be an arbitrary subset of cardinality k, and suppose that $v \in \mathcal{C}(S; \alpha)$. By definition, we have

$$\|v\|_1 = \|v_S\|_1 + \|v_{S^c}\|_1 \;\leq\; (1+\alpha)\|v_S\|_1,$$

and consequently $\|v\|_1 \leq (1+\alpha)\sqrt{k}\|v\|_2$. By assumption, we also have $\|\Sigma^{1/2} v\|_2 \geq \gamma \|v\|_2$ for all $v \in \mathcal{C}(S; \alpha)$. Substituting these two inequalities into the bound (3) yields

$$\frac{\|Xv\|_2}{\sqrt{n}} \geq \left\{ \frac{\gamma}{4} - 9\,(1+\alpha)\,\rho(\Sigma)\,\sqrt{\frac{k \log p}{n}} \right\} \|v\|_2.$$

Under the assumed scaling (4) of the sample size, we have

$$9\,(1+\alpha)\,\rho(\Sigma)\,\sqrt{\frac{k\log p}{n}} \leq \gamma/8,$$

which shows that the RE condition holds for X^TX/n with parameter $(\alpha,\gamma/8)$ as claimed. ■

Remarks:

(a) From the definitions, it is easy to see that if the RE condition holds with $\alpha = 1$ and any $\gamma > 0$ (even arbitrarily small), then the RN condition also holds. Indeed, if the matrix X^TX/n satisfies the $(1,\gamma)$-RE condition, then for any $v \in \mathcal{C}(S;1)\backslash\{0\}$, we have $\frac{\|Xv\|_2}{\sqrt{n}} \geq \gamma\|v\|_2 > 0$, which implies that $\mathcal{C}(S,1) \cap (X) = \{0\}$.

(b) As previously discussed, it is known (Bickel et al., 2009; van de Geer, 2000; van de Geer and Buhlmann, 2009) that if X^TX/n satisfies the RE condition, then the ℓ_2 error of the Lasso under the sparse linear model with Gaussian noise satisfies the bound

$$\|\widehat{\beta} - \beta^*\|_2 = O(\sqrt{\frac{k\log p}{n}}) \qquad \text{with high probability.}$$

Consequently, in order to ensure that the ℓ_2-error is bounded, the sample size must scale as $n = \Omega(k\log p)$, which matches the scaling (4) required in Corollary 1, as long as the sequence of covariance matrices Σ have diagonal entries that stay bounded.

(c) Finally, we note that Theorem 1 guarantees that the sample covariance X^TX/n satisfies a property that is slightly stronger than the RE condition. As shown by Negahban et al. (2009), this strengthening also leads to error bounds for the Lasso when β^* is not exactly k-sparse, but belongs to an ℓ_q-ball. The resulting rates are known to be minimax-optimal for these ℓ_q-balls (Raskutti et al., 2009).

3.2 Comparison to Related Work

At this point, we provide a brief comparison of our results with some related results in the literature beyond the papers discussed in the introduction. Haupt et al. (2010) showed that a certain class of random Toeplitz matrices, where the entries in the first row and first column are Bernoulli random variables and the rest fill out the Toeplitz structure satisfy RIP (and hence the weaker RE condition) provided that $n = \Omega(k^3\log(p/k))$. In Section 3.3, we demonstrate that Gausssian design matrices where the covariance matrix is a Toeplitz matrix satisfies the RE condition under the milder scaling requirement $n = \Omega(k\log(p))$. It would be of interest to determine such scaling can be established for the random Toeplitz matrices considered by Haupt et al. (2010).

It is worth comparing the scaling (4) to a related result due to van de Geer and Buhlmann (2009). In particular, their Lemma 10.1 also provides sufficient conditions for a restricted eigenvalue condition to hold for design matrices with dependent columns. They show that if the true covariance matrix satisfies an RE condition, and if the elementwise maximum $\|\widehat{\Sigma} - \Sigma\|_\infty$ between the sample covariance $\widehat{\Sigma} = X^TX/n$ and true covariance Σ is suitably bounded, then the sample covariance also satisfies the RE condition. Their result applied to the case of Gaussian random matrices guarantees

that $\widehat{\Sigma}$ satisfies the RE property as long as $n = \Omega(k^2 \log p)$ and Σ satisfies the RE condition. By contrast, Corollary 1 guarantees the RE condition with the less restrictive scaling $n = \Omega(k \log p)$. Note that if $k = O(\sqrt{n})$, our scaling condition is satisfied while their condition fails. This quadratic-linear gap in sparsity between k^2 and k arises from the difference between a local analysis (looking at individual entries of $\widehat{\Sigma}$) versus the global analysis of this paper, which studies the full random matrix. On the other hand, the result of van de Geer and Buhlmann (2009) applies more generally, including the case of sub-Gaussian random matrices (e.g., those with bounded entries) in addition to the Gaussian matrices considered in Theorem 1.

Finally, in work that followed up on the initial posting of this work (Raskutti et al., 2009), a paper by Zhou (2009) provides an extension of Theorem 1 to the case of correlated random matrices with sub-Gaussian entries. Theorem 1.6 in her paper establishes that certain families of sub-Gaussian matrices satisfy the RE condition w.h.p. with sample size $n = \Omega(s \log(p/s))$. This extension is based on techniques developed by Mendelson et al. (2008), while we use Gaussian comparison inequalities and simple concentration results for the case of Gaussian design.

3.3 Some Illustrative Examples

Let us illustrate some classes of matrices to which our theory applies. We will see that Corollary 1 applies to many sequences of covariance matrices $\Sigma = \Sigma_{p \times p}$ that have much more structure than the identity matrix. Our theory allows for the maximal eigenvalue of Σ to be arbitrarily large, or for Σ to be rank-degenerate, or for both of these degeneracies to occur at the same time. In all cases, we consider sequences of matrices for which the maximum variance $\rho^2(\Sigma) = \max_{j=1,\ldots,p} \Sigma_{jj}$ stays bounded. Under this mild restriction, we provide several examples where the RE condition is satisfied with high probability. For suitable choices, these same examples show that the RE condition can hold with high probability, even when the restricted isometry property (RIP) of Candes and Tao (2005) is violated with probability converging to one.

Example 1 (Toeplitz matrices) *Consider a covariance matrix with the Toeplitz structure $\Sigma_{ij} = a^{|i-j|}$ for some parameter $a \in [0,1)$. This type of covariance structure arises naturally from autoregressive processes, where the parameter a allows for tuning of the memory in the process. The minimum eigenvalue of Σ is $1-a > 0$, independent of the dimension p, so that the population matrix Σ clearly satisfies the RE condition. Since $\rho^2(\Sigma) = 1$, Theorem 1 implies that the sample covariance matrix $\widehat{\Sigma} = X^T X/n$ obtained by sampling from this distribution will also satisfy the RE condition with high probability as long as $n = \Omega(k \log p)$. This provides an example of a matrix family with substantial correlation between covariates for which the RE property still holds.*

However, regardless of the sample size, the submatrices of the sample covariance $\widehat{\Sigma}$ will not satisfy restricted isometry properties (RIP) if the parameter a is sufficiently large. For instance, defining $S = \{1,2,\ldots,k\}$, consider the sub-block $\widehat{\Sigma}_{SS}$ of the sample covariance matrix. Satisfying RIP requires that that the condition number $\lambda_{\max}(\widehat{\Sigma}_{SS})/\lambda_{\min}(\widehat{\Sigma}_{SS})$ be very close to one. As long as $n = \Omega(k \log p)$, known results in random matrix theory (Davidson and Szarek, 2001) guarantee that the eigenvalues of $\widehat{\Sigma}_{SS}$ will be very close to the population versions Σ_{SS}; see also the concrete calculation in Example 2 to follow. Consequently, imposing RIP amounts to requiring that the population condition number $\lambda_{\max}(\Sigma_{SS})/\lambda_{\min}(\Sigma_{SS})$ be very close to one. This condition number grows as the parameter $a \in [0,1)$ increases towards one (Gray, 1990), so RIP will be violated once $a < 1$ is sufficiently large.

We now consider a matrix family with an even higher amount of dependency among the covariates, where the RIP constants are actually unbounded as the sparsity k increases, but the RE condition is still satisfied.

Example 2 (Spiked identity model) *For a parameter $a \in [0,1)$, the spiked identity model is given by the family of covariance matrices*

$$\Sigma := (1-a) I_{p \times p} + a \vec{1}\, \vec{1}^T,$$

where $\vec{1} \in \mathbb{R}^p$ is the vector of all ones. The minimum eigenvalue of Σ is $1-a$, so that the population covariance clearly satisfies the RE condition for any fixed $a \in [0,1)$. Since this covariance matrix has maximum variance $\rho^2(\Sigma) = 1$, Corollary 1 implies that a sample covariance matrix $\widehat{\Sigma} = X^T X/n$ will satisfy the RE property with high probability with sample size $n = \Omega(k \log p)$.

On the other hand, the spiked identity matrix Σ has very poorly conditioned sub-matrices, which implies that a sample covariance matrix $\widehat{\Sigma} = X^T X/n$ will violate the restricted isometry property (RIP) with high probability as n grows. To see this fact, for an arbitrary subset S of size k, consider the associated $k \times k$ submatrix Σ_{SS}. An easy calculation shows that $\lambda_{\min}(\Sigma_{SS}) = 1-a > 0$ and $\lambda_{\max}(\Sigma_{SS}) = 1 + a(k-1)$, so that the population condition number of this sub-matrix is

$$\frac{\lambda_{\max}(\Sigma_{SS})}{\lambda_{\min}(\Sigma_{SS})} = \frac{1+a(k-1)}{1-a}.$$

For any fixed $a \in (0,1)$, this condition number diverges as k increases. We now show that the same statement applies to the sample covariance with high probability, showing that the RIP is violated. Let $u \in \mathbb{R}^k$ and $v \in \mathbb{R}^k$ denote (respectively) unit-norm eigenvectors corresponding to the minimum and maximum eigenvalues of Σ_{SS}, and define the random variables $Z_u = \|Xu\|_2^2/n$ and $Z_v = \|Xv\|_2^2/n$. Since $\langle X_i, v\rangle \sim N(0, \lambda_{\max}(\Sigma_{SS}))$ by construction, we have

$$Z_v = \frac{1}{n}\sum_{i=1}^n \langle X_i, v\rangle^2 \stackrel{d}{=} \lambda_{\max}(\Sigma_{SS}) \Big\{ \frac{1}{n}\sum_{i=1}^n y_i^2 \Big\},$$

where $y_i \sim N(0,1)$ are i.i.d. standard Gaussians, and $\stackrel{d}{=}$ denotes equality in distribution. By χ^2 tail bounds, we have $\mathbb{P}[\frac{1}{n}\sum_{i=1}^n y_i^2 \geq \frac{1}{2}] \leq c_1 \exp(-c_2 n)$, so that $Z_v \geq \lambda_{\max}(\Sigma_{SS})/2$ with high probability. A similar argument shows that $Z_u \leq 2\lambda_{\min}(\Sigma_{SS})$ with high probability, and putting together the pieces shows that w.h.p.

$$\frac{\lambda_{\max}(\widehat{\Sigma}_{SS})}{\lambda_{\min}(\widehat{\Sigma}_{SS})} \geq \frac{1}{4}\frac{\lambda_{\max}(\Sigma_{SS})}{\lambda_{\min}(\Sigma_{SS})} \geq \frac{1}{4}\,\frac{1+a(k-1)}{1-a},$$

which diverges as k increases.

In both of the preceding examples, the minimum eigenvalue of Σ was bounded from below and the diagonal entries of Σ were bounded from above, which allowed us to assert immediately that the RE condition was satisfied for the population covariance matrix. As a final example, we now consider sampling from population covariance matrices that are actually rank degenerate, but for which our theory still provides guarantees.

Example 3 (Highly degenerate covariance matrices) *Let Σ be any matrix with bounded diagonal that satisfies the RE property of some order k. Suppose that we sample n times from a $N(0,\Sigma)$ distribution, and then form the empirical covariance matrix $\widehat{\Sigma} = X^T X/n$. If $n < p$, then $\widehat{\Sigma}$ must be rank degenerate, but Corollary 1 guarantees that $\widehat{\Sigma}$ will satisfy the RE property of order k with high probability as long as $n = \Omega(k \log p)$. Moreover, by χ^2-tail bounds, the maximal diagonal element $\rho^2(\widehat{\Sigma})$ will be bounded with high probability under this same scaling.*

Now if we condition on the original design matrix X in the high probability set, we may view $\widehat{\Sigma}$ as a fixed but highly rank-degenerate matrix. Suppose that we draw a new set of n i.i.d. vectors $\widetilde{X}_i \sim N(0,\widehat{\Sigma})$ using this degenerate covariance matrix. Such a resampling procedure could be relevant for a bootstrap-type calculation for assessing errors of the Lasso. We may then form a second empirical covariance matrix $\widetilde{\Sigma} = \frac{1}{n}\widetilde{X}^T\widetilde{X}$. Conditionally on $\widehat{\Sigma}$ having the RE property of order k and a bounded diagonal, Corollary 1 shows that the resampled empirical covariance $\widetilde{\Sigma}$ will also have the RE property of order k with high probability, again for $n = \Omega(k \log p)$.

This simple example shows that in the high-dimensional setting $p \gg n$, it is possible for the RE condition to hold with high probability even when the original population covariance matrix ($\widehat{\Sigma}$ in this example) has a $p-n$-dimensional nullspace. Note moreover that this is not an isolated phenomenon—rather, it will hold for almost every sample covariance matrix $\widehat{\Sigma}$ constructed in the way that we have described.

4. Proof of Theorem 1

We now turn to the proof of Theorem 1. The main ingredients are the Gordon-Slepian comparison inequalities (Gordon, 1985) for Gaussian processes, concentration of measure for Lipschitz functions (Ledoux, 2001), and a peeling argument. The first two ingredients underlie classical proofs on the ordinary eigenvalues of Gaussian random matrices (Davidson and Szarek, 2001), whereas the latter tool is used in empirical process theory (van de Geer, 2000).

4.1 Proof Outline

Recall that Theorem 1 states that the condition

$$\frac{\|Xv\|_2}{\sqrt{n}} \geq \frac{1}{4}\|\Sigma^{1/2}v\|_2 - 9\,\rho(\Sigma)\sqrt{\frac{\log p}{n}}\,\|v\|_1 \qquad \text{for all } v \in \mathbb{R}^p, \tag{5}$$

holds with probability at least $1 - c'\exp(-cn)$, where c, c' are universal positive constants. Hence, we are bounding the random quantity $\|Xv\|_2$ in terms of $\|\Sigma^{1/2}v\|_2$ and $\|v\|_1$ for all v with high probability. It suffices to prove Theorem 1 for $\|\Sigma^{1/2}v\|_2 = 1$. Indeed, for any vector $v \in \mathbb{R}^p$ such that $\Sigma^{1/2}v = 0$, the claim holds holds trivially. Otherwise, we may consider the re-scaled vector $\check{v} = v/\|\Sigma^{1/2}v\|_2$, and note that $\|\Sigma^{1/2}\check{v}\|_2 = 1$ by construction. By scale invariance of the condition (5), if it holds for the re-scaled vector $\check{v}$, it also holds for v.

Therefore, in the remainder of the proof, our goal is to lower bound the quantity $\|Xv\|_2$ over the set of v such that $\|\Sigma^{1/2}v\|_2 = 1$ in terms of $\|v\|_1$. At a high level, there are three main steps to the proof:

(1) We begin by considering the set $V(r) := \{v \in \mathbb{R}^p \mid \|\Sigma^{1/2}v\|_2 = 1, \|v\|_1 \leq r\}$, for a fixed radius r. Although this set may be empty for certain choices of $r > 0$, our analysis only concerns

those choices for which it is non-empty. Define the random variable

$$M(r,X) := 1 - \inf_{v \in V(r)} \frac{\|Xv\|_2}{\sqrt{n}} = \sup_{v \in V(r)} \left\{ 1 - \frac{\|Xv\|_2}{\sqrt{n}} \right\}.$$

Our first step is to upper bound the expectation $\mathbb{E}[M(r,X)]$, where the expectation is taken over the random Gaussian matrix X.

(2) Second, we establish that $M(r,X)$ is a Lipschitz function of its Gaussian arguments, and then use concentration inequalities to assert that for each fixed $r > 0$, the random variable $M(r,X)$ is sharply concentrated around its expectation with high probability.

(3) Third, we use a peeling argument to show that our analysis holds with high probability and uniformly over all possible choice of the ℓ_1-radius r, which then implies that the condition (5) holds with high probability as claimed.

In the remainder of this section, we provide the details of each of these steps.

4.2 Bounding the Expectation $\mathbb{E}[M(r,X)]$

This subsection is devoted to a proof of the following lemma:

Lemma 1 *For any radius $r > 0$ such that $V(r)$ is non-empty, we have*

$$\mathbb{E}[M(r,X)] \leq \frac{1}{4} + 3\rho(\Sigma)\sqrt{\frac{\log p}{n}}\, r.$$

Proof: Let $S^{n-1} = \{u \in \mathbb{R}^n \mid \|u\|_2 = 1\}$ be the Euclidean sphere of radius 1, and recall the previously defined set $V(r) := \{v \in \mathbb{R}^p \mid \|\Sigma^{1/2} v\|_2 = 1, \|v\|_1 \leq r\}$. For each pair $(u,v) \in S^{n-1} \times V(r)$, we may define an associated zero-mean Gaussian random variable $Y_{u,v} := u^T X v$. This representation is useful, because it allows us to write the quantity of interest as a min-max problem in terms of this Gaussian process. In particular, we have

$$-\inf_{v \in V(r)} \|Xv\|_2 = -\inf_{v \in V(r)} \sup_{u \in S^{n-1}} u^T X v = \sup_{v \in V(r)} \inf_{u \in S^{n-1}} u^T X v.$$

We may now upper bound the expected value of the above quantity via a Gaussian comparison inequality; here we state a form of Gordon's inequality used in past work on Gaussian random matrices (Davidson and Szarek, 2001). Suppose that $\{Y_{u,v}, (u,v) \in U \times V\}$ and $\{Z_{u,v}, (u,v) \in U \times V\}$ are two zero-mean Gaussian processes on $U \times V$. Using $\sigma(\cdot)$ to denote the standard deviation of its argument, suppose that these two processes satisfy the inequality

$$\sigma(Y_{u,v} - Y_{u',v'}) \leq \sigma(Z_{u,v} - Z_{u',v'}) \qquad \text{for all pairs } (u,v) \text{ and } (u',v') \text{ in } U \times V,$$

and this inequality holds with equality when $v = v'$. Then we are guaranteed that

$$\mathbb{E}[\sup_{v \in V} \inf_{u \in U} Y_{u,v}] \quad \leq \quad \mathbb{E}[\sup_{v \in V} \inf_{u \in U} Z_{u,v}].$$

We use Gordon's inequality to show that

$$\mathbb{E}[M(r,X)] = 1 + \mathbb{E}[\sup_{v \in V(r)} \inf_{u \in S^{n-1}} Y_{u,v}] \leq 1 + \mathbb{E}[\sup_{v \in V(r)} \inf_{u \in S^{n-1}} Z_{u,v}],$$

where we recall that $Y_{u,v} = u^T X v$ and $Z_{u,v}$ is a different Gaussian process to be defined shortly.

We begin by computing $\sigma^2(Y_{u,v} - Y_{u',v'})$. To simplify notation, we note that the $X \in \mathbb{R}^{n\times p}$ can be written as $W\Sigma^{1/2}$, where $W \in \mathbb{R}^{n\times p}$ is a matrix with i.i.d. $\mathcal{N}(0,1)$ entries, and $\Sigma^{1/2}$ is the symmetric matrix square root. In terms of W, we can write

$$Y_{u,v} = u^T W \Sigma^{1/2} v \;=\; u^T W \tilde{v},$$

where $\tilde{v} = \Sigma^{1/2} v$. It follows that

$$\sigma^2(Y_{u,v} - Y_{u',v'}) := \mathbb{E}\big(\sum_{i=1}^n \sum_{j=1}^p W_{i,j}(u_i\tilde{v}_j - u'_i\tilde{v}'_j)\big)^2 \;=\; |||u\tilde{v}^T - (u')(\tilde{v}')^T|||_F^2,$$

where $|||\cdot|||_F$ is the Frobenius norm (ℓ_2-norm applied elementwise to the matrix). This equality follows immediately since the W_{ij} variables are i.i.d $\mathcal{N}(0,1)$.

Now consider a second zero-mean Gaussian process $Z_{u,v}$ indexed by $S^{n-1} \times V(r)$, and given by

$$Z_{u,v} \;\;=\;\; \vec{g}^T u + \vec{h}^T \Sigma^{1/2} v,$$

where $\vec{g} \sim N(0, I_{n\times n})$ and $\vec{h} \sim N(0, I_{p\times p})$ are standard Gaussian random vectors. With $\tilde{v} = \Sigma^{1/2} v$, we see immediately that

$$\sigma^2(Z_{u,v} - Z_{u',v'}) = \|u - u'\|_2^2 + \|\tilde{v} - \tilde{v}'\|_2^2.$$

Consequently, in order to apply the Gaussian comparison principle to $\{Y_{u,v}\}$ and $\{Z_{u,v}\}$, we need to show that

$$|||u\tilde{v}^T - (u')(\tilde{v}')^T|||_F^2 \leq \|u - u'\|_2^2 + \|\tilde{v} - \tilde{v}'\|_2^2 \tag{6}$$

for all pairs $(u, \tilde{v})$ and $(u', \tilde{v}')$ in the set of interest. Since the Frobenius norm $|||\cdot|||_F$ is simply the ℓ_2-norm on the vectorized form of a matrix, we can compute

$$\begin{aligned}
|||u\tilde{v}^T - u'(\tilde{v}')^T|||_F^2 &= |||(u-u')\tilde{v}^T + u'(\tilde{v} - \tilde{v}')^T|||_F^2 \\
&= \sum_{i=1}^n \sum_{j=1}^p [(u_i - u'_i)\tilde{v}_j + u'_i(\tilde{v}_j - \tilde{v}'_j)]^2 \\
&= \|\tilde{v}\|_2^2 \|u - u'\|_2^2 + \|u'\|_2^2 \|\tilde{v} - \tilde{v}'\|_2^2 + 2(u^T u' - \|u'\|_2^2)(\|\tilde{v}\|_2^2 - \tilde{v}^T\tilde{v}') \\
&= \|u - u'\|_2^2 + \|\tilde{v} - \tilde{v}'\|_2^2 - 2(\|u'\|_2^2 - u^T u')(\|\tilde{v}\|_2^2 - \tilde{v}^T\tilde{v}'),
\end{aligned}$$

where we have used equalities $\|u\|_2 = \|u'\|_2 = 1$ and $\|\tilde{v}\|_2 = \|\tilde{v}'\|_2 = 1$. By the Cauchy-Schwarz inequality, we have $\|u\|_2^2 - u^T u' \geq 0$, and $\|\tilde{v}\|_2^2 - \tilde{v}^T\tilde{v}' \geq 0$, from which the claimed inequality (6) follows. When $v = v'$, we also have $\tilde{v} = \Sigma^{1/2} v = \Sigma^{1/2} v' = \tilde{v}'$, so that equality holds in the condition (6) when $\tilde{v} = \tilde{v}'$.

Consequently, we may apply Gordon's inequality to conclude that

$$\begin{aligned}
\mathbb{E}\big[\sup_{v\in V(r)} \inf_{u\in S^{n-1}} u^T X v\big] &\leq \mathbb{E}\big[\sup_{v\in V(r)} \inf_{u\in S^{n-1}} Z_{u,v}\big] \\
&= \mathbb{E}[\inf_{u\in S^{n-1}} \vec{g}^T u] + \mathbb{E}[\sup_{v\in V(r)} \vec{h}^T \Sigma^{1/2} v] \\
&= -\mathbb{E}[\|\vec{g}\|_2] + \mathbb{E}[\sup_{v\in V(r)} \vec{h}^T \Sigma^{1/2} v].
\end{aligned}$$

We now observe that by definition of $V(r)$, we have

$$\sup_{v \in V(r)} |\vec{h}^T \Sigma^{1/2} v| \leq \sup_{v \in V(r)} \|v\|_1 \, \|\Sigma^{1/2}\vec{h}\|_\infty \leq r \|\Sigma^{1/2}\vec{h}\|_\infty.$$

Each element $(\Sigma^{1/2}\vec{h})_j$ is zero-mean Gaussian with variance Σ_{jj}. Consequently, known results on Gaussian maxima (cf. Ledoux and Talagrand, 1991, Equation (3.13)) imply that $\mathbb{E}[\|\Sigma^{1/2}h\|_\infty] \leq 3\sqrt{\rho^2(\Sigma)\log p}$, where $\rho^2(\Sigma) = \max_j \Sigma_{jj}$. Noting[2] that $\mathbb{E}[\|\vec{g}\|_2] \geq \frac{3}{4}\sqrt{n}$ for all $n \geq 10$ by standard χ^2 tail bounds and putting together the pieces, we obtain the bound

$$\mathbb{E}[-\inf_{v \in V(r)} \|Xv\|_2] \quad \leq \quad -\frac{3}{4}\sqrt{n} + 3\big[\rho^2(\Sigma)\log p\big]^{1/2} r.$$

Dividing by $\sqrt{n}$ and adding 1 to both sides yields

$$\mathbb{E}[M(r,X)] = \mathbb{E}[1 - \inf_{v \in V(r)} \|Xv\|_2/\sqrt{n}] \leq 1/4 + 3\rho(\Sigma)\sqrt{\frac{\log p}{n}}\, r,$$

as claimed.

■

4.3 Concentration Around the Mean for $M(r,X)$

Having controlled the expectation, the next step is to establish concentration of $M(r,X)$ around its mean. Note that Lemma 1 shows that $\mathbb{E}[M(r,X)] \leq t(r)$, where

$$t(r) := \frac{1}{4} + 3r\,\rho(\Sigma)\sqrt{\frac{\log p}{n}}. \tag{7}$$

Now we prove the following claim:

Lemma 2 *For any r such that $V(r)$ is non-empty, we have*

$$\mathbb{P}\left[M(r,X) \geq \frac{3t(r)}{2}\right] \leq 2\exp(-nt^2(r)/8).$$

Proof In order to prove this lemma, it suffices to show that

$$\mathbb{P}\big[|M(r,X) - \mathbb{E}[M(r,X)]| \geq t(r)/2\big] \leq 2\exp(-nt^2(r)/8),$$

and use the upper bound on $\mathbb{E}[M(r,X)]$ derived in Lemma 1.

By concentration of measure for Lipschitz functions of Gaussians (see Appendix B), this tail bound will follow if we show that the Lipschitz constant of $M(r,X)$ as a function of the Gaussian random matrix is less than $1/\sqrt{n}$. To make this functional dependence explicit, let us write $M(r,X)$ as the function $h(W) = \sup_{v \in V(r)} \big(1 - \|W\Sigma^{1/2}v\|_2/\sqrt{n}\big)$. We find that

$$\sqrt{n}\big[h(W) - h(W')\big] = \sup_{v \in V(r)} -\|W\Sigma^{1/2}v\|_2 - \sup_{v \in V(r)} -\|W'\Sigma^{1/2}v\|_2.$$

2. In fact, $|\mathbb{E}[\|\vec{g}\|_2] - \sqrt{n}| = o(\sqrt{n})$, but this simple bound is sufficient for our purposes.

Since $V(r)$ is closed and bounded and the objective function is continuous, there exists $\hat{v} \in V(r)$ such that $\hat{v} = \arg\max_{v \in V(r)} -\|W\Sigma^{1/2}v\|_2$. Therefore

$$\begin{aligned}\sup_{v\in V(r)} \big(-\|W\Sigma^{1/2}v\|_2\big) - \sup_{v\in V(r)} \big(-\|W'\Sigma^{1/2}v\|_2\big) &= -\|W\Sigma^{1/2}\hat{v}\|_2 - \sup_{v\in V(r)} \big(-\|W'\Sigma^{1/2}v\|_2\big) \\ &\leq \|W'\Sigma^{1/2}\hat{v}\|_2 - \|W\Sigma^{1/2}\hat{v}\|_2 \\ &\leq \sup_{v\in V(r)} \big(\|(W'-W)\Sigma^{1/2}v\|_2\big).\end{aligned}$$

For a matrix A, we define its spectral norm $|||A|||_2 = \sup_{\|u\|_2=1} \|Au\|_2$. With this notation, we can bound the Lipschitz constant of h as

$$\begin{aligned}\sqrt{n}\big[h(W) - h(W')\big] &\leq \sup_{v\in V(r)} \big(\|(W-W')\Sigma^{1/2}v\|_2\big) \\ &\overset{(a)}{\leq} \big\{\sup_{v\in V(r)} \big(\|\Sigma^{1/2}v\|_2\big)\big\} |||(W-W')|||_2 \\ &\overset{(b)}{\leq} \big\{\sup_{v\in V(r)} \big(\|\Sigma^{1/2}v\|_2\big)\big\}\ |||(W-W')|||_F \\ &\overset{(c)}{=} |||W-W'|||_F.\end{aligned}$$

In this argument, inequality (a) follows by definition of the matrix spectral norm $|||\cdot|||_2$; inequality (b) follows from the bound $|||(W-W')|||_2 \leq |||(W-W')|||_F$ between the spectral and Frobenius matrix norms (Horn and Johnson, 1985); and equality (c) follows since $\|\Sigma^{1/2}v\|_2 = 1$ for all $v \in V(r)$. Thus, we have shown that h has Lipschitz constant $L \leq 1/\sqrt{n}$ with respect to the Euclidean norm on W (viewed as a vector with np entries). Finally we use a standard result on the concentration for Lipschitz functions of Gaussian random variables (Ledoux, 2001; Massart, 2003)—see Appendix B for one statement. Applying the concentration result (9) with $m = np$, $\tilde{g} = W$, and $t = t(r)/2$ completes the proof. ■

4.4 Extension to All Vectors Via Peeling

Thus far, we have shown that

$$M(r,X) = 1 - \inf_{v\in V(r)} \frac{\|Xv\|_2}{\sqrt{n}} = \sup_{v\in V(r)} \left\{1 - \frac{\|Xv\|_2}{\sqrt{n}}\right\} \geq 3t(r)/2, \tag{8}$$

with probability no larger than $2\exp(-nt^2(r)/8)$ where $t(r) = \frac{1}{4} + 3\,r\,\rho(\Sigma)\sqrt{\frac{\log p}{n}}$. The set $V(r)$ requires that $\|v\|_1 \leq r$ for some *fixed* radius r, whereas the claim of Theorem 1 applies to all vectors v. Consequently, we need to extend the bound (8) to an arbitrary ℓ_1 radius.

In order do so, we define the event

$$\mathcal{T} := \big\{\exists\, v \in \mathbb{R}^p \text{ s.t. } \|\Sigma^{1/2}v\|_2 = 1 \text{ and } \big(1 - \|Xv\|_2/\sqrt{n}\big) \geq 3t(\|v\|_1)\big\}.$$

Note that there is no r in the definition of $\mathcal{T}$, because we are setting $\|v\|_1$ to be the argument of the function t. We claim that there are constants positive constants c, c' such that $\mathbb{P}[\mathcal{T}] \leq c\exp(-c'n)$,

from which Theorem 1 will follow. We establish this claim by using a device known as peeling (Alexander, 1985; van de Geer, 2000); for the version used here, see Lemma 3 proved in the Appendix. In particular, we apply Lemma 3 with the functions

$$f(v,X) = 1 - \|Xv\|_2/\sqrt{n}, \qquad h(v) = \|v\|_1, \quad \text{and} \quad g(r) = 3t(r)/2,$$

the sequence $a_n = n$, and the set $A = \{v \in \mathbb{R}^p \mid \|\Sigma^{1/2}v\|_2 = 1\}$. Recall that the quantity t, as previously defined (7), satisfies $t(r) \geq 1/4$ for all $r > 0$ and is strictly increasing. Therefore, the function $g(r) = 3t(r)/2$ is non-negative and strictly increasing as a function of r, and moreover satisfies $g(r) \geq 3/8$, so that Lemma 3 is applicable with $\mu = 3/8$. We can thus conclude that $\mathbb{P}[\mathcal{T}^c] \geq 1 - c\exp(-c'n)$ for some numerical constants c and c'.

Finally, conditioned on the event $\mathcal{T}^c$, for all $v \in \mathbb{R}^p$ with $\|\Sigma^{1/2}v\|_2 = 1$, we have

$$1 - \|Xv\|_2/\sqrt{n} \;\leq\; 3t(\|v\|_1) \;=\; \frac{3}{4} + 9\,\|v\|_1\,\rho(\Sigma)\sqrt{\frac{\log p}{n}},$$

which implies that

$$\|Xv\|_2/\sqrt{n} \geq \frac{1}{4} - 9\,\|v\|_1\,\rho(\Sigma)\,\sqrt{\frac{\log p}{n}}.$$

As noted in the proof outline, this suffices to establish the general claim.

5. Conclusion

Methods based on ℓ_1-relaxations are very popular, and the weakest possible conditions on the design matrix X required to provide performance guarantees—namely, the restricted nullspace and eigenvalue conditions—are well-understood. In this paper, we have proved that these conditions hold with high probability for a broad class of Gaussian design matrices allowing for quite general dependency among the columns, as captured by a covariance matrix Σ representing the dependence among the different covariates. As a corollary, our result guarantees that known performance guarantees for ℓ_1-relaxations such as basis pursuit and Lasso hold with high probability for such problems, provided the population matrix Σ satisfies the RE condition. Interestingly, our theory shows that ℓ_1-methods can perform well when the covariates are sampled from a Gaussian distribution with a degenerate covariance matrix. Some follow-up work (Zhou, 2009) has extended these results to random matrices with sub-Gaussian rows. In addition, there are a number of other ways in which this work could be extended. One is to incorporate additional dependence across the rows of the design matrix, as would arise in modeling time series data for example. It would also be interesting to relate the allowable degeneracy structures of Σ to applications involving real data. Finally, although this paper provides various conditions under which the RE condition holds with high probability, it does not address the issue of how to determine whether a given sample covariance matrix matrix $\widehat{\Sigma} = X^T X/n$ satisfies the RE condition. It would be interesting to study if there are computationally efficient methods for verifying the RE condition.

Acknowledgments

We thank Arash Amini for useful discussion, particularly regarding the proofs of Theorem 1 and Lemma 3, and Rob Nowak for helpful comments on an earlier draft. This work was partially

supported by NSF grants DMS-0605165 and DMS-0907632 to MJW and BY. In addition, BY was partially supported by the NSF grant SES-0835531 (CDI), and a grant from the MSRA. MJW was supported by an Sloan Foundation Fellowship and AFOSR Grant FA9550-09-1-0466. During this work, GR was financially supported by a Berkeley Graduate Fellowship.

Appendix A. Peeling Argument

In this appendix, we state a result on large deviations of the constrained optimum of random objective functions of the form $f(v;X)$, where $v \in \mathbb{R}^p$ is the vector to be optimized over, and X is some random vector. Of interest is the problem $\sup_{h(v)\leq r,\, v\in A} f(v;X)$, where $h : \mathbb{R}^p \to \mathbb{R}_+$ is some non-negative and increasing constraint function, and A is a non-empty set. With this set-up, our goal is to bound the probability of the event defined by

$$\mathcal{E} \quad := \quad \big\{\exists\, v \in A \text{ such that } f(v;X) \geq 2g(h(v)))\big\},$$

where $g : \mathbb{R} \to \mathbb{R}$ is non-negative and strictly increasing.

Lemma 3 *Suppose that $g(r) \geq \mu$ for all $r \geq 0$, and that there exists some constant $c > 0$ such that for all $r > 0$, we have the tail bound*

$$\mathbb{P}\big[\sup_{v\in A,\, h(v)\leq r} f(v;X) \geq g(r)\big] \quad \leq \quad 2\exp(-c\, a_n\, g^2(r)),$$

for some $a_n > 0$. Then we have

$$\mathbb{P}[\mathcal{E}] \quad \leq \quad \frac{2\exp(-4ca_n\mu^2)}{1-\exp(-4ca_n\mu^2)}.$$

Proof : Our proof is based on a standard peeling technique (e.g., see van de Geer, 2000, p. 82). By assumption, as v varies over A, we have $g(r) \in [\mu, \infty)$. Accordingly, for $m = 1, 2, \ldots$, defining the sets

$$A_m \quad := \quad \big\{v \in A \mid 2^{m-1}\mu \leq g(h(v)) \leq 2^m\mu\big\},$$

we may conclude that if there exists $v \in A$ such that $f(v,X) \geq 2g(h(v))$, then this must occur for some m and $v \in A_m$. By union bound, we have

$$\mathbb{P}[\mathcal{E}] \quad \leq \quad \sum_{m=1}^{\infty} \mathbb{P}\big[\exists\, v \in A_m \text{ such that } f(v,X) \geq 2g(h(v))\big].$$

If $v \in A_m$ and $f(v,X) \geq 2g(h(v))$, then by definition of A_m, we have $f(v,X) \geq 2\,(2^{m-1})\mu = 2^m\mu$. Since for any $v \in A_m$, we have $g(h(v)) \leq 2^m\mu$, we combine these inequalities to obtain

$$\begin{aligned}
\mathbb{P}[\mathcal{E}] \quad &\leq \quad \sum_{m=1}^{\infty} \mathbb{P}\big[\sup_{h(v)\leq g^{-1}(2^m\mu)} f(v,X) \geq 2^m\mu\big] \\
&\leq \quad \sum_{m=1}^{\infty} 2\exp\big(-ca_n\,[g(g^{-1}(2^m\mu))]^2\big) \\
&= \quad 2\sum_{m=1}^{\infty} \exp\big(-ca_n\,2^{2m}\mu^2\big),
\end{aligned}$$

from which the stated claim follows by upper bounding this geometric sum. ■

Appendix B. Concentration for Gaussian Lipschitz Functions

We say that a function $F : \mathbb{R}^m \to \mathbb{R}$ is Lipschitz with constant L if $|F(x) - F(y)| \leq L\|x - y\|_2$ for all $x, y \in \mathbb{R}^m$. It is a classical fact that Lipschitz functions of standard Gaussian vectors exhibit Gaussian concentration. We summarize one version of this fact in the following:

Theorem 2 (Theorem 3.8 from Massart 2003) *Let $w \sim \mathcal{N}(0, I_{m\times m})$ be an m-dimensional Gaussian random variable. Then for any L-Lipschitz function F, we have*

$$\mathbb{P}\left[|F(w) - \mathbb{E}[F(w)]| \geq t\right] \leq 2\exp\left(-\frac{t^2}{2L^2}\right), \text{ for all } t \geq 0. \tag{9}$$

This result can be interpreted as saying that in terms of tail behavior, the random variable $F(w) - \mathbb{E}[F(w)]$ behaves like a zero-mean Gaussian with variance L^2.

References

R. Adamczak, A. Litvak, N. Tomczak-Jaegermann, and A. Pajor. Restricted isometry property of matrices with independent columns and neighborly polytopes by random sampling. Technical report, University of Alberta, 2009.

K. S. Alexander. Rates of growth for weighted empirical processes. In *Proceedings of the Berkeley Conference in Honor of Jerzy Neyman and Jack Kiefer*, pages 475–493. UC Press, Berkeley, 1985.

P. Bickel, Y. Ritov, and A. Tsybakov. Simultaneous analysis of lasso and dantzig selector. *Annals of Statistics*, 37(4):1705–1732, 2009.

F. Bunea, A. Tsybakov, and M. Wegkamp. Sparsity oracle inequalities for the Lasso. *Electronic Journal of Statistics*, pages 169–194, 2007.

E. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. Info Theory*, 51(12):4203–4215, December 2005.

E. Candes and T. Tao. The Dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics*, 35(6):2313–2351, 2007.

S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Computing*, 20(1):33–61, 1998.

A. Cohen, W. Dahmen, and R. DeVore. Compressed sensing and best k-term approximation. *J. of. American Mathematical Society*, 22(1):211–231, January 2009.

K. R. Davidson and S. J. Szarek. Local operator theory, random matrices, and Banach spaces. In *Handbook of Banach Spaces*, volume 1, pages 317–336. Elsevier, Amsterdan, NL, 2001.

D. Donoho. Compressed sensing. *IEEE Trans. Info. Theory*, 52(4):1289–1306, April 2006.

D. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Trans. Info Theory*, 47(7):2845–2862, 2001.

M. Elad and A. M. Bruckstein. A generalized uncertainty principle and sparse representation in pairs of bases. *IEEE Trans. Info Theory*, 48(9):2558–2567, September 2002.

A. Feuer and A. Nemirovski. On sparse representation in pairs of bases. *IEEE Trans. Info Theory*, 49(6):1579–1581, 2003.

Y. Gordon. Some inequalities for Gaussian processes and applications. *Israel Journal of Mathematics*, 50(4):265–289, 1985.

R. M. Gray. Toeplitz and Circulant Matrices: A Review. Technical report, Stanford University, Information Systems Laboratory, 1990.

O. Guédon, S. Mendelson, A. Pajor, and N. Tomczak-Jaegermann. Subspaces and orthogonal decompositions generated by bounded orthogonal systems. *Journal of Positivity*, 11(2):269–283, 2007.

O. Guédon, S. Mendelson, A. Pajor, and N. Tomczak-Jaegermann. Majorizing measures and proportional subsets of bounded orthonormal systems. *Journal of Rev. Mat. Iberoam*, 24(3):1075–1095, 2008.

J. Haupt, W. U. Bajwa, G. Raz, and R. Nowak. Toeplitz compressed sensing matrices with applications to sparse channel estimation. Technical report, University of Wisconsin-Madison, 2010.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.

M. Ledoux. *The Concentration of Measure Phenomenon*. Mathematical Surveys and Monographs. American Mathematical Society, Providence, RI, 2001.

M. Ledoux and M. Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer-Verlag, New York, NY, 1991.

P. Massart. *Concentration Inequalties and Model Selection*. Ecole d'Eté de Probabilités, Saint-Flour. Springer, New York, 2003.

N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for high-dimensional data. *Annals of Statistics*, 37(1):246–270, 2009.

S. Mendelson, A. Pajor, and N. Tomczak-Jaegermann. Uniform uncertainty principle for bernoulli and subgaussian ensembles. *Journal of Constr. Approx.*, 28(3):277–289, 2008.

S. Negahban, P. Ravikumar, M. J. Wainwright, and B. Yu. A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers. In *Proceedings of NIPS*, December 2009.

G. Raskutti, M. J. Wainwright, and B. Yu. Minimax rates of estimation for high-dimensional linear regression over ℓ_q-balls. Technical report, U. C. Berkeley, October 2009. Posted as http://arxiv.org/abs/0910.2042.

Justin Romberg. Compressive sensing by random convolution. *SIAM Journal of Imaging Science*, 2(4):1098–1128, 2009.

M. Rudelson and R. Vershynin. On sparse reconstruction from fourier and gaussian measurements. *Comm. Pure and Appl. Math.*, 61(8):1025–1045, 2008.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

S. van de Geer. *Empirical Processes in M-Estimation*. Cambridge University Press, 2000.

S. van de Geer. The deterministic lasso. In *Proc. of Joint Statistical Meeting*, 2007.

S. van de Geer and P. Buhlmann. On the conditions used to prove oracle results for the lasso. *Electronic Journal of Statistics*, 3:1360–1392, 2009.

C. H. Zhang and J. Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *Annals of Statistics*, 36(4):1567–1594, 2008.

S. Zhou. Restricted eigenvalue conditions on subgaussian random matrices. Technical report, Department of Mathematics, ETH Zürich, December 2009.

Journal of Machine Learning Research 11 (2010) 2261-2286 Submitted 12/09; Revised 4/10; Published 7/10

High Dimensional Inverse Covariance Matrix Estimation via Linear Programming

Ming Yuan MYUAN@ISYE.GATECH.EDU
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205, USA

Editor: John Lafferty

Abstract

This paper considers the problem of estimating a high dimensional inverse covariance matrix that can be well approximated by "sparse" matrices. Taking advantage of the connection between multivariate linear regression and entries of the inverse covariance matrix, we propose an estimating procedure that can effectively exploit such "sparsity". The proposed method can be computed using linear programming and therefore has the potential to be used in very high dimensional problems. Oracle inequalities are established for the estimation error in terms of several operator norms, showing that the method is adaptive to different types of sparsity of the problem.

Keywords: covariance selection, Dantzig selector, Gaussian graphical model, inverse covariance matrix, Lasso, linear programming, oracle inequality, sparsity

1. Introduction

One of the classical problems in multivariate statistics is to estimate the covariance matrix or its inverse. Let $X = (X_1, \ldots, X_p)'$ be a p-dimensional random vector with an unknown covariance matrix Σ_0. The goal is to estimate Σ_0 or its inverse $\Omega_0 := \Sigma_0^{-1}$ based on n independent copies of X, $X^{(1)}, \ldots, X^{(n)}$. The usual sample covariance matrix is most often adopted for this purpose:

$$S = \frac{1}{n}\sum_{i=1}^{n}(X^{(i)} - \bar{X})(X^{(i)} - \bar{X})',$$

where $\bar{X} = \sum X^{(i)}/n$. The behavior of S is well understood and it is known to perform well in the classical setting when the dimensionality p is small (see, e.g., Anderson, 2003; Muirhead, 2005). On the other hand, with the recent advances in science and technology, we are more and more often faced with the problem of high dimensional covariance matrix estimation where the dimensionality p is large when compared with the sample size n. Given the large number of parameters ($p(p+1)/2$) involved, exploiting the sparse nature of the problem becomes critical. In particular, traditional estimates such as S do not take advantage of the possible sparsity and are known to perform poorly under many usual matrix norms when p is large. Motivated by the practical demands and the failure of classical methods, a number of sparse models and approaches have been introduced in recent years to deal with high dimensional covariance matrix estimation. See, for example, Ledoit and Wolf (2004), Levina, Rothman and Zhu (2007), Deng and Yuan (2008), El Karoui (2008), Fan, Fan and Lv (2008), Ravikumar, Raskutti, Wainwright and Yu (2008), Ravikumar, Wainwright, Ruskutti

and Yu (2008), Rocha, Zhao and Yu (2008), Lam and Fan (2009), and Rothman, Levina and Zhu (2009) among others.

Bickel and Levina (2008a) pioneered the theoretical study of high dimensional sparse covariance matrices. They consider the case where the magnitude of the entries of Σ_0 decays at a polynomial rate of their distance from the diagonal; and show that banding the sample covariance matrix or S leads to well-behaved estimates. More recently, Cai, Zhang and Zhou (2010) established minimax convergence rates for estimating this type of covariance matrices. A more general class of covariance matrix model is investigated in Bickel and Levina (2008b) where the rows or columns of Σ_0 is assumed to come from an ℓ_α ball with $0 < \alpha < 1$. They suggest thresholding the entries of S and study its theoretical behavior when p is large. In addition to the aforementioned methods, sparse models have also been proposed for the modified Cholesky factor of the covariance matrix in a series of papers by Pourahmadi and co-authors (Pourahmadi, 1999; Pourahmadi, 2000; Wu and Pourahmadi, 2003; Huang et al., 2006).

In this paper, we focus on another type of sparsity—sparsity in terms of the entries of the inverse covariance matrix. This type of sparsity naturally connects with the problem of covariance selection (Dempster, 1972) and Gaussian graphical models (see, e.g., Whittaker, 1990; Lauritzen, 1996; Edwards, 2000), which makes it particularly appealing in a number of applications. Methods to exploit such sparsity have been proposed recently. Inspired by the nonnegative garrote (Breiman, 1995) and Lasso (Tibshirani, 1996) for the linear regression, Yuan and Lin (2007) propose to impose ℓ_1 type of penalty on the entries of the inverse covariance matrix when maximizing the normal log-likelihood and therefore encourages some of the entries of the estimated inverse covariance matrix to be exact zero. Similar approaches are also taken by Banerjee, El Ghaoui and d'Aspremont (2008). One of the main challenges for this type of methods is computation which has been recently addressed by d'Aspremont, Banerjee and El Ghaoui (2008), Friedman, Hastie and Tibshirani (2008), Rocha, Zhao and Yu (2008), Rothman et al. (2008) and Yuan (2008). Some theoretical properties of this type of methods have also been developed by Yuan and Lin (2007), Ravikumar et al. (2008), Rothman et al. (2008) and Lam and Fan (2009) among others. In particular, the results from Ravikumar et al. (2008) and Rothman et al. (2008) suggest that, although better than the sample covariance matrix, these methods may not perform well when p is larger than the sample size n. It remains unclear to what extent the sparsity of inverse covariance matrix entails well-behaved covariance matrix estimates.

Through the study of a new estimating procedure, we show here that the estimability of a high dimensional inverse covariance matrix is related to how well it can be approximated by a graphical model with a relatively low degree. The revelation that the degree of a graph dictates the difficulty of estimating a high dimensional covariance matrix suggests that the proposed method may be more appropriate to harness sparsity in the inverse covariance matrix than those mentioned earlier in which the ℓ_1 penalty serves as a proxy to control the total number of edges in the graph as opposed to its degree. The proposed method proceeds in two steps. A preliminary estimate is first constructed using a well known relationship between inverse covariance matrix and multivariate linear regression. We show that the preliminary estimate, although often dismissed as an estimate of the inverse covariance matrix, can be easily modified to produce a satisfactory estimate for the inverse covariance matrix. We show that the resulting estimate enjoys very good theoretical properties by establishing oracle inequalities for the estimation error.

The probabilistic bounds we prove suggest that the estimation error of the proposed method adapts to the sparseness of the true inverse covariance matrix. The implications of these oracle in-

equalities are demonstrated on a couple of popular covariance matrix models. When Ω_0 corresponds to a Gaussian graphical model of degree d, we show that the proposed method can achieve convergence rate of the order $O_p[d(n^{-1}\log p)^{-1/2}]$ in terms of several matrix operator norms. We also examine the more general case where the rows or columns of Ω_0 belong to an ℓ_α ball ($0 < \alpha < 1$), the family of positive definite matrices introduced by Bickel and Levina (2008b). We show that the proposed method achieves the convergence rate of $O_p[(n^{-1}\log p)^{(1-\alpha)/2}]$, the same as that obtained by Bickel and Levina (2008b) when assuming that Σ_0 rather than Ω_0 belongs to the same family of matrices. For both examples, we also show that the obtained rates are optimal in a minimax sense when considering estimation error in terms of matrix ℓ_1 or ℓ_∞ norms.

The proposed method shares similar spirits with the neighborhood selection approach proposed by Meinshausen and Bühlmann (2006). However, the two techniques are developed for different purposes. Neighborhood selection aims at identifying the correct graphical model whereas our goal is to estimate the covariance matrix. The distinction is clear when the inverse covariance matrix is only "approximately" sparse and does not have many zero entries. Even when the inverse covariance matrix is indeed sparse, the two tasks of estimation and selection can be different. In particular, our results suggest that good estimation can be achieved under conditions weaker than those often assumed to ensure good selection.

The rest of the paper is organized as follows. In the next section, we describe in details the estimating procedure. Theoretical properties of the method are established in Section 3. All detailed proofs are relegated to Section 6. Numerical experiments are presented in Section 4 to illustrate the merits of the proposed method before concluding with some remarks in Section 5.

2. Methodology

In what follows, we shall write $X_{-i} = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_p)'$. Similarly, denote by $\Sigma_{-i,-j}$ the submatrix of Σ with its ith row and jth column removed. Other notation can also be interpreted in the same fashion. For example, $\Sigma_{i,-j}$ or $\Sigma_{-i,j}$ represents the ith row of Σ with its j entry removed or the jth column with its ith entry removed respectively.

2.1 Regression and Inverse Covariance Matrix

It is well known that if X follows a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$, then the conditional distribution of X_i given X_{-i} remains normally distributed (Anderson, 2003), that is,

$$X_i | X_{-i} \sim \mathcal{N}\left(\mu_i + \Sigma_{i,-i}\Sigma_{-i,-i}^{-1}(X_{-i} - \mu_{-i}), \Sigma_{ii} - \Sigma_{i,-i}\Sigma_{-i,-i}^{-1}\Sigma_{-i,i}\right).$$

This can be equivalently expressed as the following regression equation:

$$X_i = \alpha_i + X_{-i}'\theta_{(i)} + \varepsilon_i, \tag{1}$$

where $\alpha_i = \mu_i - \Sigma_{i,-i}\Sigma_{-i,-i}^{-1}\mu_{-i}$ is a scalar, $\theta_{(i)} = \Sigma_{-i,-i}^{-1}\Sigma_{-i,i}$ is a $p-1$ dimensional vector and $\varepsilon_i \sim \mathcal{N}(0, \Sigma_{ii} - \Sigma_{i,-i}\Sigma_{-i,-i}^{-1}\Sigma_{-i,i})$ is independent of X_{-i}. When X follows a more general distribution, similar relationship holds in that $\alpha_i + X_{-i}'\theta_{(i)}$ is the best linear unbiased estimate of X_i given X_{-i} whereas $\mathrm{Var}(\varepsilon_i) = \Sigma_{ii} - \Sigma_{i,-i}\Sigma_{-i,-i}^{-1}\Sigma_{-i,i}$.

Now by the inverse formula for block matrices, $\Omega := \Sigma^{-1}$ is given by

$$\begin{pmatrix} \Sigma_{11} & \Sigma_{1,-1} \\ \Sigma_{-1,1} & \Sigma_{-1,-1} \end{pmatrix}^{-1} = \begin{pmatrix} \overbrace{\left(\Sigma_{11} - \Sigma_{1,-1}\Sigma_{-1,-1}^{-1}\Sigma_{-1,1}\right)^{-1}}^{\Omega_{11}} & -\Omega_{11}\Sigma_{1,-1}\Sigma_{-1,-1}^{-1} \\ -\Sigma_{-1,-1}^{-1}\Sigma_{-1,1}\Omega_{11} & * \end{pmatrix}.$$

More generally, the ith column of Ω can be written as

$$\begin{aligned} \Omega_{ii} &= \left(\Sigma_{ii} - \Sigma_{i,-i}\Sigma_{-i,-i}^{-1}\Sigma_{-i,i}\right)^{-1}; \\ \Omega_{-i,i} &= -\left(\Sigma_{ii} - \Sigma_{i,-i}\Sigma_{-i,-i}^{-1}\Sigma_{-i,i}\right)^{-1}\Sigma_{-i,-i}^{-1}\Sigma_{-i,i}. \end{aligned}$$

This immediately connects with (1):

$$\begin{aligned} \Omega_{ii} &= (\mathrm{Var}(\varepsilon_i))^{-1}; \\ \Omega_{-i,i} &= -(\mathrm{Var}(\varepsilon_i))^{-1}\theta_{(i)}. \end{aligned}$$

Therefore, an estimate of Ω can potentially be obtained by regressing X_i over X_{-i} for $i = 1, \ldots, p$. Furthermore, the sparsity in the entries of Ω can be translated into sparsity in regression coefficients $\theta_{(i)}$s.

2.2 Initial Estimate

From the aforementioned relationship, a zero entry on the ith column of the inverse covariance matrix implies a zero entry in the regression coefficient $\theta_{(i)}$ and vice versa. This property is exploited by Meinshausen and Bühlmann (2006) to identify the zero pattern of the inverse covariance matrix. Specifically, in the so-called neighborhood selection method, the zero entries of the ith column of Ω_0 are identified by doing variable selection when regressing X_i over X_{-i}. More specifically, they suggest to use Lasso (Tibshirani, 1996) for the purpose of variable selection.

Our goal here, however, is rather different. Instead of identifying which entries of Ω_0 are zero, our focus is on estimating it. The distinction is apparent when Ω_0 is only "approximately" sparse instead of having a lot of zero entries. Even if Ω_0 indeed has lot of zeros, the two tasks can still be quite different in high dimensional problems. For example, in identifying the nonzero entries of Ω_0, it is necessary to assume that all nonzero entries are sufficiently different from zero (see, e.g., Meinshausen and Bühlmann, 2006). Such assumptions may be unrealistic and can be relaxed if the purpose is to estimate the covariance matrix. With such a distinction in mind, the question now is whether or not similar strategies of applying sparse multivariate linear regression to recover the inverse covariance matrix remains useful. The answer is affirmative.

To this end, we consider estimating Ω_0 as follows:

$$\begin{aligned} \tilde{\Omega}_{ii} &= \left(\widehat{\mathrm{Var}}(\varepsilon_i)\right)^{-1}; \\ \tilde{\Omega}_{-i,i} &= -\left(\widehat{\mathrm{Var}}(\varepsilon_i)\right)^{-1}\hat{\theta}_{(i)}, \end{aligned}$$

where $\widehat{\mathrm{Var}}(\varepsilon_i)$ and $\hat{\theta}_{(i)}$ are estimated from regressing X_i over X_{-i}. In particular, we suggest to use the so-called Dantzig selector (Candès and Tao, 2007) for estimating the regression coefficients. We

begin by centering each variable X_i to eliminate the intercept α_i in (1). Denote by $Z_i = X_i - \bar{X}_i$ where $\bar{X}_i$ is the sample average of X_i. The Dantzig selector estimate of $\theta_{(i)}$ is the solution to

$$\min_{\beta \in \mathbb{R}^{p-1}, \beta_0 \in \mathbb{R}} \|\beta\|_{\ell_1} \qquad \text{subject to } \left\| \mathbb{E}_n \left[\left(Z_i - Z'_{-i}\beta \right) Z_{-i} \right] \right\|_{\ell_\infty} \leq \delta,$$

where $\mathbb{E}_n$ represents the sample average, and $\delta > 0$ is a tuning parameter. Recall that $\mathbb{E}_n Z_i Z_j = S_{ij}$. The above problem can also be written in terms of S:

$$\min_{\beta \in \mathbb{R}^{p-1}, \beta_0 \in \mathbb{R}} \|\beta\|_{\ell_1} \qquad \text{subject to } \left\| S_{-i,i} - S_{-i,-i}\beta \right\|_{\ell_\infty} \leq \delta. \tag{2}$$

The minimization of the ℓ_1 norm of the regression coefficient reflects our preference towards sparse models which is particularly important when dealing with high dimensional problems. Once an estimate of $\theta_{(i)}$ is obtained, we can then estimate the variance of ε_i by the mean squared error of the residuals:

$$\widehat{\mathrm{Var}}(\varepsilon_i) = \mathbb{E}_n \left(X_i - X'_{-i}\hat{\theta}_{(i)} \right)^2 = S_{ii} - 2\hat{\theta}'_{(i)} S_{-i,i} + \hat{\theta}'_{(i)} S_{-i,-i} \hat{\theta}_{(i)}.$$

We obtain $\tilde{\Omega}$ by repeating this procedure for $i = 1, \ldots, p$.

We emphasize that for practical purposes, one can also use the Lasso in place of the Dantzig selector for constructing $\tilde{\Omega}$. The choice of Dantzig selector is made for the sake of our further technical developments. In the light of the results of Bickel, Ritov and Tsybakov (2009), similar performance can be expected with either the Lasso or the Dantzig selector although a more rigorous proof when using the Lasso is beyond the scope of the current paper.

2.3 Symmetrization

$\tilde{\Omega}$ is is usually dismissed as an estimate of Ω for it is not even symmetric. In fact, it is not obvious that $\tilde{\Omega}$ is in any sense a reasonable estimate of Ω_0. But a more careful examination suggests otherwise. It reveals that $\tilde{\Omega}$ could be a good estimate in a certain matrix operator norm.

The matrix operator norm is a class of matrix norms induced by vector norms. Let $\|\mathbf{x}\|_{\ell_q}$ be the ℓ_q norm of an p dimensional vector $\mathbf{x} = (x_1, \ldots, x_p)'$, that is,

$$\|\mathbf{x}\|_{\ell_q} = \left(|x_1|^q + \ldots + |x_p|^q \right)^{1/q}.$$

Then the matrix ℓ_q norm for an $p \times p$ square matrix $A = (a_{ij})_{1 \leq i,j \leq p}$ is given by

$$\|A\|_{\ell_q} = \sup_{\mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|_{\ell_q}}{\|\mathbf{x}\|_{\ell_q}}.$$

In the case of $q = 1$ and $q = \infty$, the matrix norm can be given more explicitly as

$$\begin{aligned} \|A\|_{\ell_1} &= \max_{1 \leq j \leq p} \sum_{i=1}^{p} \left| a_{ij} \right|; \\ \|A\|_{\ell_\infty} &= \max_{1 \leq i \leq p} \sum_{j=1}^{p} \left| a_{ij} \right|. \end{aligned}$$

When $q = 2$, the matrix operator norm of A amounts to its leading singular value, and is often referred to as the spectral norm.

A careful study shows that $\tilde{\Omega}$ can be a good estimate of Ω in terms of the matrix ℓ_1 norm in sparse circumstances. It is therefore of interest to consider improved estimates from $\tilde{\Omega}$ that inherits this property. To this end, we propose to adjust $\tilde{\Omega}$ by seeking a symmetric matrix $\hat{\Omega}$ that is the the closest to $\tilde{\Omega}$ in the sense of the matrix ℓ_1 norm, that is, it solves the following problem:

$$\min_{\Omega \text{ is symmetric}} \|\Omega - \tilde{\Omega}\|_{\ell_1}. \tag{3}$$

Recall that

$$\|\Omega - \tilde{\Omega}\|_{\ell_1} = \max_{1 \le j \le p} \sum_{i=1}^{p} \left|\Omega_{ij} - \tilde{\Omega}_{ij}\right|.$$

Problem (3) can therefore be re-formulated as a linear program just like the computation of $\tilde{\Omega}$.

To sum up, our estimate of the inverse covariance matrix is obtained in the following steps:

ALGORITHM FOR COMPUTING $\hat{\Omega}$

Input: Sample covariance matrix – S, tuning parameter – δ.
Output: An estimate of the inverse covariance matrix – $\hat{\Omega}$.

- **Construct $\tilde{\Omega}$**
  for $i = 1$ to p
  - Estimate $\theta_{(i)}$ by $\hat{\theta}_{(i)}$, the solution to

$$\min_{\beta \in \mathbb{R}^{p-1}} \|\beta\|_{\ell_1} \qquad \text{subject to } \left\|S_{-i,i} - S_{-i,-i}\beta\right\|_{\ell_\infty} \le \delta.$$

  - Set

$$\tilde{\Omega}_{ii} = \left(S_{ii} - 2\hat{\theta}'_{(i)}S_{-i,i} + \hat{\theta}'_{(i)}S_{-i,-i}\hat{\theta}_{(i)}\right)^{-1}.$$

  - Set

$$\tilde{\Omega}_{-i,i} = -\tilde{\Omega}_{ii}\hat{\theta}_{(i)}.$$

  end

- **Construct $\hat{\Omega}$**
  - Set $\hat{\Omega}$ as the solution to

$$\min_{\Omega \text{ is symmetric}} \|\Omega - \tilde{\Omega}\|_{\ell_1}.$$

It is worth pointing out that that proposed method depends on the data only through the sample covariance matrix. This fact is of great practical importance since it suggests that a large sample size will not affect the computational complexity in calculating $\hat{\Omega}$ more than the evaluation of S. Furthermore, only linear programs are involved in the computation of $\hat{\Omega}$, which makes the approach appealing when dealing with very high dimensional problems.

3. Theory

In what follows, we shall assume that the components of X are uniformly sub-gaussian, that is, there exist constants $c_0 \geq 0$, and $T > 0$ such that for any $|t| \leq T$

$$\mathbb{E}e^{tX_i^2} \leq c_0, \qquad i = 1, 2, \ldots, p.$$

This condition is clearly satisfied when X follows a multivariate normal distribution. It also holds true when X_is are bounded.

3.1 Oracle Inequality

Our main tool to study the theoretical properties of $\hat{\Omega}$ is an oracle type of inequality regarding the estimation error $\|\hat{\Omega} - \Omega_0\|_{\ell_1}$. To this end, we introduce the following set of "oracle" inverse covariance matrices:

$$O(\nu, \eta, \tau) = \left\{ \Omega \succ 0 : \begin{array}{ll} \nu^{-1} \leq \lambda_{\min}(\Omega) \leq \lambda_{\max}(\Omega) \leq \nu & \text{(Bounded Eigenvalues)} \\ \|\Sigma_0 \Omega - I\|_{\max} \leq \eta & \text{(“Good” Approximation)} \\ \|\Omega\|_{\ell_1} \leq \tau & \text{(Sparsity)} \end{array} \right\},$$

where $A \succ 0$ indicates that a matrix A is symmetric and positive definite; $\nu > 1, \tau > 0$, and $\eta \geq 0$ are parameters; $\lambda_{\min}$ and $\lambda_{\max}$ represent the smallest and largest eigenvalue respectively; and $\|\cdot\|_{\max}$ represents the entry-wise ℓ_∞ norm, that is,

$$\|A\|_{\max} = \max_{1 \leq i,j \leq p} |a_{ij}|.$$

We refer to $O(\nu, \eta, \tau)$ as an "oracle" set because its definition requires the knowledge of the true covariance matrix Σ_0. Every member of $O(\nu, \eta, \tau)$ is symmetric, positive definite with eigenvalues bounded away from 0 and ∞, and belongs to an ℓ_1 ball. Moreover, $O(\nu, \eta, \tau)$ consists of matrices that approximate Ω_0 well. It is worth noting that different from the usual vector case, the choice of metric is critical when evaluating approximating error for matrices. In particular for our purpose, the approximation error is measured by $\|\Sigma_0 \Omega - I\|_{\max}$, which vanishes if and only if $\Omega = \Omega_0$. We are now in position to state our main result.

Theorem 1 *There exist constants C_1, C_2 depending only on ν, τ, $\lambda_{\min}(\Omega_0)$ and $\lambda_{\max}(\Omega_0)$, and C_3 depending only on c_0 such that, for any $A > 0$, with probability at least $1 - p^{-A}$,*

$$\left\|\hat{\Omega} - \Omega_0\right\|_{\ell_1} \leq C_1 \inf_{\Omega \in O(\nu,\eta,\tau)} \left(\left\|\Omega - \Omega_0\right\|_{\ell_1} + \deg(\Omega)\delta \right), \tag{4}$$

provided that

$$\inf_{\Omega \in O(\nu,\eta,\tau)} \left(\left\|\Omega - \Omega_0\right\|_{\ell_1} + \deg(\Omega)\delta \right) \leq C_2, \tag{5}$$

and

$$\delta \geq \nu\eta + C_3 \nu\tau \lambda_{\min}^{-1}(\Omega_0)((A+1)n^{-1}\log p)^{1/2}, \tag{6}$$

where $\deg(\Omega) = \max_i \sum_j \mathbb{I}(\Omega_{ij} \neq 0)$.

We remark that the oracle inequality given in Theorem 1 is of probabilistic nature and non-asymptotic. However, (4) holds with overwhelming probability as we are interested in the case when p is very large. The requirement (5) is in place to ensure that the true inverse covariance matrix is indeed "approximately" sparse. Another note is on the choice of the tuning parameter δ. To ensure an tight upper bound in (4), smaller δs are preferred. On the other hand, Condition (6) specifies how small they can be. For simplicity, we have used the same tuning parameter δ for estimating all $\theta_{(i)}$s. In practice, it may be beneficial to use different δs for different $\theta_{(i)}$s. Following the same argument, it can be shown that the statement of Theorem 1 continue to hold if all tuning parameters used satisfy Condition (6).

Recall that for a symmetric matrix A, $\|A\|_{\ell_\infty} = \|A\|_{\ell_1}$ and

$$\|A\|_{\ell_2} \le (\|A\|_{\ell_1}\|A\|_{\ell_\infty})^{1/2} = \|A\|_{\ell_1}.$$

A direct consequence of Theorem 1 is that the same upper bound holds true under matrix ℓ_∞ and ℓ_2 norms.

Corollary 2 *There exist constants C_1, C_2 depending only on ν, τ, $\lambda_{\min}(\Omega_0)$ and $\lambda_{\max}(\Omega_0)$, and C_3 depending only on c_0 such that, for any $A > 0$, with probability at least $1 - p^{-A}$,*

$$\left\|\hat{\Omega} - \Omega_0\right\|_{\ell_\infty}, \left\|\hat{\Omega} - \Omega_0\right\|_{\ell_2} \le C_1 \inf_{\Omega \in O(\nu,\eta,\tau)} \left(\left\|\Omega - \Omega_0\right\|_{\ell_1} + \deg(\Omega)\delta\right),$$

provided that

$$\inf_{\Omega \in O(\nu,\eta,\tau)} \left(\left\|\Omega - \Omega_0\right\|_{\ell_1} + \deg(\Omega)\delta\right) \le C_2,$$

and

$$\delta \ge \nu\eta + C_3\nu\tau\lambda_{\min}^{-1}(\Omega_0)((A+1)n^{-1}\log p)^{1/2}.$$

The bound on the matrix ℓ_2 has great practical implications when we are interested in estimating the covariance matrix or need to a positive definite estimate of Ω. The proposed estimate $\hat{\Omega}$ is symmetric but not guaranteed to be positive definite. However, Corollary 2 suggests that with overwhelming probability, it is indeed positive definite provided that the upper bound is sufficiently small because

$$\lambda_{\min}(\hat{\Omega}) \ge \lambda_{\min}(\Omega_0) - \left\|\hat{\Omega} - \Omega_0\right\|_{\ell_2}.$$

Moreover, a positive definite estimate of Ω can always be constructed by replacing its negative eigenvalues with δ. Denote the resulting estimate by $\hat{\hat{\Omega}}$. By Corollary 2, it can be shown that

Corollary 3 *There exist constants C_1, C_2 depending only on ν, τ, $\lambda_{\min}(\Omega_0)$ and $\lambda_{\max}(\Omega_0)$, and C_3 depending only on c_0 such that, for any $A > 0$, with probability at least $1 - p^{-A}$,*

$$\left\|\hat{\hat{\Omega}}^{-1} - \Sigma_0\right\|_{\ell_2}, \left\|\hat{\hat{\Omega}} - \Omega_0\right\|_{\ell_2} \le C_1 \inf_{\Omega \in O(\nu,\eta,\tau)} \left(\left\|\Omega - \Omega_0\right\|_{\ell_1} + \deg(\Omega)\delta\right),$$

provided that

$$\inf_{\Omega \in O(\nu,\eta,\tau)} \left(\left\|\Omega - \Omega_0\right\|_{\ell_1} + \deg(\Omega)\delta\right) \le C_2,$$

and

$$\delta \ge \nu\eta + C_3\nu\tau\lambda_{\min}^{-1}(\Omega_0)((A+1)n^{-1}\log p)^{1/2}.$$

When considering a particular class of inverse covariance matrices, we can use the oracle inequalities established here with a proper choice of the oracle set O. Typically in choosing a good oracle set O, we take ν and τ to be of finite magnitude whereas the approximation error η sufficiently small. To further illustrate their practical implications, we now turn to a couple of more concrete examples.

3.2 Sparse Models

We begin with a class of matrix models that are closely connected with graphical models. When X follows a multivariate normal distribution, the sparsity of the entries of the inverse covariance matrix relates to the notion of conditional independence: the (i, j) entry of Ω_0 being zero implies that X_i is independent of X_j conditional on the remaining variables and vice versa. The conditional independence relationships among the coordinates of the Gaussian random vector X can be represented by an undirected graph $G = (V, E)$, often referred to as a Gaussian graphical model, where V contains p vertices corresponding to the p coordinates and the edge between X_i and X_j is present if and only if X_i and X_j are not independent conditional on the others. The complexity of a graphical model is commonly measured by its degree:

$$\deg(G) = \max_{1 \le i \le p} \sum_j e_{ij},$$

where $e_{ij} = 1$ if there is an edge between X_i and X_j and 0 otherwise. Gaussian graphical models are an indispensable statistical tool in studying communication networks and gene pathways among many other subjects. The readers are referred to Whittaker (1990), Lauritzen (1996) and Edwards (2000) for further details.

Motivated by this connection, we consider the following class of inverse covariance matrices:

$$\mathcal{M}_1(\tau_0, \nu_0, d) = \left\{ A \succ 0 : \|A\|_{\ell_1} < \tau_0, \nu_0^{-1} < \lambda_{\min}(A) < \lambda_{\max}(A) < \nu_0, \deg(A) < d \right\},$$

where $\tau_0, \nu_0 > 1$, and $\deg(A) = \max_i \sum_j \mathbb{I}(A_{ij} \neq 0)$. In this case, taking an oracle set such that $\Omega_0 \in O$ yields the following result:

Theorem 4 *Assume that $d(n^{-1} \log p)^{1/2} = o(1)$. Then*

$$\sup_{\Omega_0 \in \mathcal{M}_1(\tau_0, \nu_0, d)} \left\| \hat{\Omega} - \Omega_0 \right\|_{\ell_q} = O_p\left(d \sqrt{\frac{\log p}{n}} \right), \tag{7}$$

provided that $\delta = C(n^{-1} \log p)^{1/2}$ and C is large enough.

Theorem 4 follows immediately from Theorem 1 and Corollary 2 by taking $\eta = 0, \tau = \|\Omega_0\|_{\ell_1}$, and $\nu = \max\{\lambda_{\min}^{-1}(\Omega_0), \lambda_{\max}(\Omega_0)\}$, which ensures that $\Omega_0 \in O(\nu, \eta, \tau)$. We note that the rate of convergence given by (7) is also optimal in the minimax sense when considering matrix ℓ_1 norm.

Theorem 5 *Assume that $d(n^{-1} \log p)^{1/2} = o(1)$. Then there exists a constant $C > 0$ depending only on τ_0, and ν_0 such that*

$$\inf_{\bar{\Omega}} \sup_{\Omega_0 \in \mathcal{M}_1(\tau_0, \nu_0, d)} \mathbb{P}\left\{ \left\| \bar{\Omega} - \Omega_0 \right\|_{\ell_1} \ge C d \sqrt{\frac{\log p}{n}} \right\} > 0,$$

where the infimum is taken over all estimate $\bar{\Omega}$ based on observations $X^{(1)}, \ldots, X^{(n)}$.

Theorem 5 indicates that the estimability of a sparse inverse covariance matrix is dictated by its degree as opposed to the total number of nonzero entries. This observation gives a plausible explanation on why the usual ℓ_1 penalized likelihood estimate (see, e.g., Yuan and Lin, 2007; Banarjee, El Ghaoui and d'Aspremont, 2008) may not be the best to exploit this type of sparsity because the penalty employed by these methods is convex relaxations of the constraint on total number of edges in a graphical model instead of its degree.

It is also of interest to compare our results with those from Meinshausen and Bühlmann (2006). As mentioned before, the goal of the neighborhood selection from Meinshausen and Bühlmann (2006) is to select the correct graphical model whereas our focus here is on estimating the covariance matrix. However, the neighborhood selection method can be followed by the maximum likelihood estimation based on the selected graphical model to yield a covariance matrix estimate. Clearly the success of this method hinges upon the ability of the neighborhood selection to choose a correct graphical model. It turns out that selecting the graphical model can be more difficult than estimating the covariance matrix as reflected by the more restrictive assumptions made in Meinshausen and Bühlmann (2006). In particular, to be able to identify the nonzero entries of the inverse covariance matrix, it is necessary that they are sufficiently large in magnitude whereas such requirement is generally not needed for the purpose of estimation. Moreover, Meinshausen and Bühlmann (2006) only deals with the case when the dimensionality is of a polynomial order of the sample size, that is, $p = O(n^\gamma)$ for some $\gamma > 0$.

3.3 Approximately Sparse Models

In many applications, the inverse covariance matrix is only approximately sparse. A popular way to model this class of covariance matrix is to assume that its rows or columns belong to an ℓ_α ball ($0 < \alpha < 1$):

$$\mathcal{M}_2(\tau_0, \nu_0, \alpha, M) = \left\{ A \succ 0 : \|A^{-1}\|_{\ell_1} < \tau_0, \nu_0^{-1} \le \lambda_{\min}(A) \le \lambda_{\max}(A) \le \nu_0, \sum_{j=1}^{p} \left|A_{ij}\right|^\alpha \le M \right\},$$

where $\tau_0, \nu_0 > 1$ and $0 < \alpha < 1$. $\mathcal{M}_2$ can be viewed as a natural extension of the sparse model $\mathcal{M}_1$. In particular, $\mathcal{M}_1$ can be viewed as the limiting case of $\mathcal{M}_2$ when α approaches 0. By relaxing α, $\mathcal{M}_2$ includes matrices that are less sparse than those included in $\mathcal{M}_1$. The particular class of matrices were first introduced by Bickel and Levina (2008b) who investigate the case when $\Sigma_0 \in \mathcal{M}_2(\tau_0, \nu_0, \alpha, M)$. We note that their setting is different from ours as $\mathcal{M}_2$ is not closed with respect to inversion. An application of Theorem 1 and Corollary 2 yields:

Theorem 6 *Assume that* $M\left(n^{-1}\log p\right)^{\frac{1-\alpha}{2}} = o(1)$. *Then*

$$\sup_{\Omega_0 \in \mathcal{M}_2(\tau_0, \nu_0, \alpha, M)} \|\hat{\Omega} - \Omega_0\|_{\ell_q} = O_p\left(M \left(\frac{\log p}{n} \right)^{\frac{1-\alpha}{2}} \right), \tag{8}$$

provided that $\delta = C(n^{-1}\log p)^{1/2}$ *and* C *is sufficiently large*.

Assuming that $\Sigma_0 \in \mathcal{M}_2$, Bickel and Levina (2008b) study thresholding estimator of the covariance matrix. Their setting is different from ours because $\mathcal{M}_2$ is not closed under inversion. It is

however interesting to note that Bickel and Levina (2008b) show that thresholding the sample covariance matrix S at an appropriate level can achieve the same rate given by right hand side of (8). The coincidence should not come as a surprise despite the difference in problem setting because the size of the parameter space in both problems are the same. Moreover, the following theorem shows that in both settings, the rate is optimal in the minimax sense.

Theorem 7 *Assume that* $M\left(n^{-1}\log p\right)^{\frac{1-\alpha}{2}} = o(1)$. *Then there exists a constant* $C > 0$ *depending only on* τ_0, *and* ν_0 *such that*

$$\inf_{\bar{\Omega}} \sup_{\Omega_0 \in \mathcal{M}_2(\tau_0,\nu_0,\alpha,M)} \mathbb{P}\left\{ \left\|\bar{\Omega} - \Omega_0\right\|_{\ell_1} \geq CM\left(\frac{\log p}{n}\right)^{\frac{1-\alpha}{2}} \right\} > 0, \tag{9}$$

and

$$\inf_{\bar{\Sigma}} \sup_{\Sigma_0 \in \mathcal{M}_2(\tau_0,\nu_0,\alpha,M)} \mathbb{P}\left\{ \left\|\bar{\Sigma} - \Sigma_0\right\|_{\ell_1} \geq CM\left(\frac{\log p}{n}\right)^{\frac{1-\alpha}{2}} \right\} > 0, \tag{10}$$

where the infimum is taken over all estimate, $\bar{\Omega}$ *or* $\bar{\Sigma}$, *based on observations* $X^{(1)},\ldots,X^{(n)}$.

4. Numerical Experiments

To illustrate the merits of the proposed method and compare it with other popular alternatives, we now conduct a set of numerical studies. Specifically, we generated $n = 50$ observations from a multivariate normal distribution with mean 0 and variance covariance matrix given by $\Sigma^0_{ij} = \rho^{|i-j|}$ for some $\rho \neq 0$. Such covariance structure corresponds to an AR(1) model. Its inverse covariance matrix is banded with the magnitude of ρ determining the strength of the dependence among the coordinates. We consider combinations of seven different values of $\rho, 0.1, 0.2,\ldots, 0.7$ and four values of the dimensionality, $p = 25, 50, 100$ or 200. Two hundred data sets were simulated for each combination. For each simulated data set, we ran the proposed method to construct estimate of the inverse covariance matrix. As suggested by the theoretical developments, we set $\delta = (2n^{-1}\log p)^{-1}$ throughout all simulation studies. For comparison purposes, we included a couple of popular alternative covariance matrix estimates in the study. The first is the ℓ_1 penalized likelihood estimate of Yuan and Lin (2007). As suggested by Yuan and Lin (2007), the BIC criterion was used to choose the tuning parameter among a total of 20 pre-specified values. The second is a variant of the neighborhood selection approach of Meinshausen and Bühlmann (2006). As pointed out earlier, the goal of the neighborhood selection is to identify the underlying graphical model rather than estimating the covariance matrix. We consider here a simple two-step procedure where the maximum likelihood estimate based on the selected graphical model is employed. As advocated by Meinshausen and Bühlmann (2006), the level of significance is set at $\alpha = 0.05$ in identifying the graphical model. Figure 1 summarizes the estimation error measured by the spectral norm, that is, $\|\hat{C} - C\|_{\ell_2}$, for the three methods, averaged over two hundred runs.

A few observations can be made from Figure 1. We first note that the proposed method tends to outperform the other two methods when ρ is small and the advantage becomes more evident as the dimensionality increases. On the other hand, the advantage over the neighborhood selection based method gradually vanishes as ρ increases yet the proposed method remains competitive. A plausible explanation is the distinction between estimation and selection in high dimensional problems as pointed out earlier. The success of the neighbor selection based method hinges upon a good selection

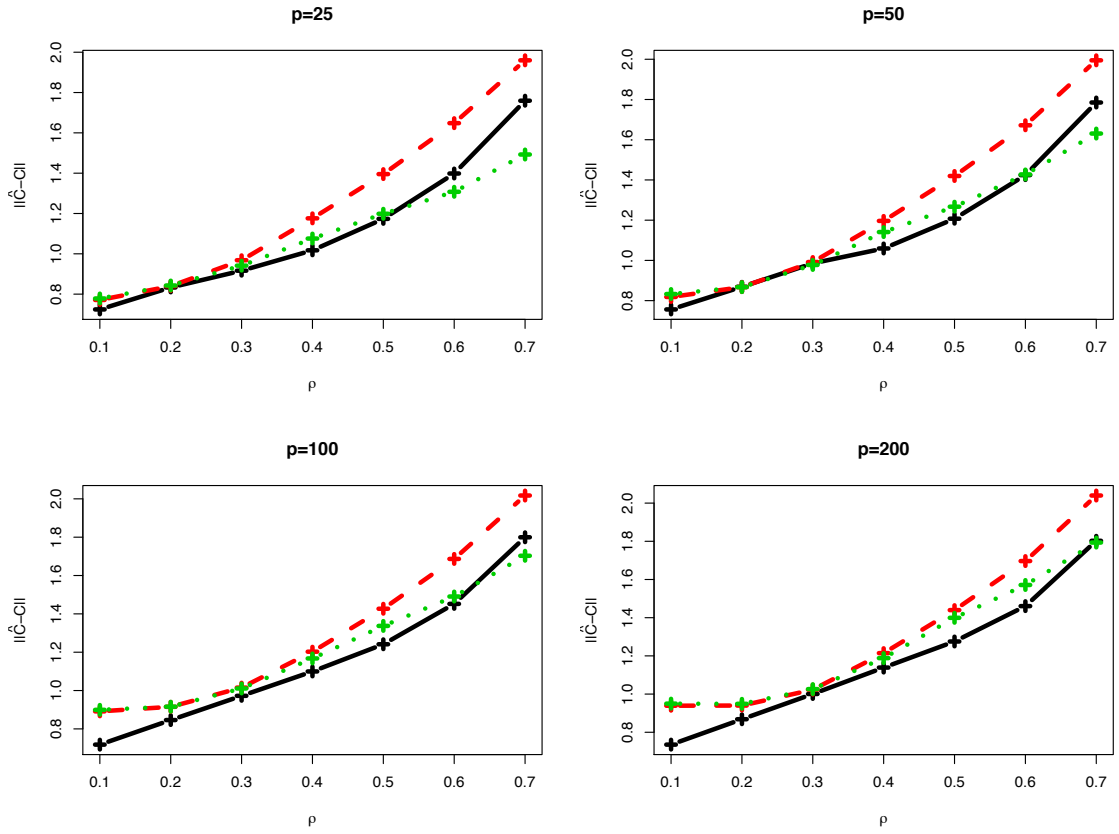


Figure 1: Estimation error of the proposed method (black solid lines), the ℓ_1 penalized likelihood estimate (red dashed lines) and the maximum likelihood estimate based on the graphical model selected through neighborhood selection (green dotted lines). Each panel corresponds to a different value of the dimensionality. X-axes represent the value of ρ. The estimation errors are averaged over two hundred runs.

of the graphical model. Recall that the inverse covariance matrix is banded with nonzero entries increasing in magnitude with ρ. For large values of ρ, the task of identifying the correct graphical model is relatively easier. With a good graphical model chosen, refitting it with the maximum likelihood estimator could reduce biases often associated with regularization approaches. Such benefit diminishes for small values of ρ as identifying nonzero entries in the inverse covariance matrix becomes more difficult.

At last, we note that all three methods are relatively efficient to compute. For example, when $p = 200$ and $\rho = 0.5$, the averaged CPU time for the ℓ_1 penalized likelihood estimate is 0.53 seconds, for the neighborhood selection based method is 1.42 seconds, and for the proposed method is 3.21 seconds. Both the ℓ_1 penalized likelihood estimate and the neighborhood selection based method are computed using the graphical Lasso algorithm of Friedman, Hastie and Tibshirani (2008) which iteratively solves a sequence of p Lasso problems using a modified Lars algorithm (Efron et al., 2004). The algorithm is specially developed to take advantage of the sparse nature of the problem

and available in the `glasso` package in R. The proposed method is implemented in MATLAB using its general purpose interior-point algorithm based linear programming solver and could be further improved using more specialized algorithms (see, e.g., Asif, 2008).

5. Discussions

High dimensional (inverse) covariance matrix estimation is becoming more and more common in various scientific and technological areas. Most of the existing methods are designed to benefit from sparsity of the covariance matrix, and based on banding or thresholding the sample covariance matrix. Sparse models for the inverse covariance matrix, despite its practical appeal and close connection to graphical modeling, are more difficult to be taken advantage of due to heavy computational cost as well as the lack of a coherent theory on how such sparsity can be effectively exploited. In this paper, we propose an estimating procedure that addresses both challenges. The proposed method can be formulated using linear programming and therefore computed very efficiently. We also show that the resulting estimate enjoys nice probabilistic properties, which translates to sharp convergence rates in terms of matrix operator norms under a couple of common settings.

The choice of the tuning parameter δ is of great practical importance. Our theoretical developments have suggested reasonable choices of the tuning parameter and it seems to work well in the well-controlled simulation settings. In practice, however, a data-driven choice such as those determined by multi-fold cross-validation may yield improved performance.

We also note that the method can be easily extended to handle prior information regarding the sparsity patterns of the inverse covariance matrices. Such situations often arise in the context of, for example, genomics. In a typical gene expression experiment, tens of thousands of genes are often studied simultaneously. Among these genes, there are often known pathways which corresponding to conditional (in)dependence among a subset of the genes, or in our notation, variables. This can be naturally interpreted as some of the entries of Ω_0 being known to be nonzero or zero. Such prior information can be easily incorporated in our procedure. In particular, it suffices to set some of the entries of β to be exact zero apriori in (2). Likewise, if a particular entry of β is known to be nonzero, we can also opt to minimize the ℓ_1 norm of only the remaining entries.

6. Proofs

We now present the proofs to Theorems 1, 5, 6 and 7.

6.1 Proof of Theorem 1

We begin by comparing $\hat{\theta}_{(i)}$ with $\theta_{(i)}$. For brevity, we shall abbreviate the subscript (i) in what follows when no confusion occurs. Recall that $\theta = -\Omega^0_{-i,i}/\Omega^0_{ii}$ and

$$\hat{\theta} = \underset{\beta \in \mathcal{F}}{\operatorname{argmin}} \|\beta\|_{\ell_1},$$

where $\mathcal{F} = \{\beta : \|S_{-i,i} - S_{-i,-i}\beta\|_{\ell_\infty} \le \delta\}$. For a given $\Omega \in O(\nu, \eta, \tau)$, let $\Omega \in O$ and $\gamma = -\Omega_{-i,i}/\Omega_{ii}$. We first show that $\gamma \in \mathcal{F}$.

Lemma 8 *Under the event that* $\|S - \Sigma_0\|_{\max} < C_0 \lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}$,

$$\|S_{-i,i} - S_{-i,-i}\gamma\|_{\ell_\infty} \le \delta,$$

provided that

$$\delta \geq \eta\nu + C_0\tau\nu\lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}.$$

Proof By the definition of $O(\nu,\eta,\tau)$, for any $j \neq i$,

$$\left|\Sigma^0_{j\cdot}\Omega_{\cdot i}\right| = \Omega_{ii}\left|\Sigma^0_{ji} - \Sigma^0_{j,-i}\gamma\right| \leq \|\Sigma_0\Omega - I\|_{\max} \leq \eta,$$

which implies that

$$\left\|\Sigma^0_{-i,i} - \Sigma^0_{-i,-i}\gamma\right\|_{\ell_\infty} = \max_{j\neq i}\left|\Sigma^0_{ji} - \Sigma^0_{j,-i}\gamma\right| \leq \Omega^{-1}_{ii}\eta \leq \lambda^{-1}_{\min}(\Omega)\eta \leq \eta\nu.$$

An application of the triangular inequality now yields

$$\begin{aligned}
\|S_{-i,i} - S_{-i,-i}\gamma\|_{\ell_\infty} &\leq \left\|S_{-i,i} - \Sigma^0_{-i,i}\right\|_{\ell_\infty} + \left\|\left(S_{-i,-i} - \Sigma^0_{-i,-i}\right)\gamma\right\|_{\ell_\infty} + \left\|\Sigma^0_{-i,i} - \Sigma^0_{-i,-i}\gamma\right\|_{\ell_\infty} \\
&\leq \|S - \Sigma_0\|_{\max} + \|S - \Sigma_0\|_{\max}\|\gamma\|_{\ell_1} + \eta\nu \\
&= \|S - \Sigma_0\|_{\max}\|\Omega_{\cdot i}\|_{\ell_1}/\Omega_{ii} + \eta\nu \\
&\leq \tau\nu\|S - \Sigma_0\|_{\max} + \eta\nu.
\end{aligned}$$

The claim now follows. ■

Now that $\gamma \in \mathcal{F}$, by the definition of $\hat{\theta}$,

$$\|\hat{\theta}\|_{\ell_1} \leq \|\gamma\|_{\ell_1} \leq \Omega^{-1}_{ii}\|\Omega_{\cdot i}\|_{\ell_1} - 1 \leq \lambda^{-1}_{\min}(\Omega)\|\Omega\|_{\ell_1} - 1 \leq \nu\tau - 1. \tag{11}$$

Write $\mathcal{J} = \{j : \gamma_j \neq 0\}$. Denote by $d_{\mathcal{J}} = \mathrm{card}(\mathcal{J})$. It is clear that $d_{\mathcal{J}} \leq \deg(\Omega)$. From (11),

$$0 \leq \|\gamma\|_{\ell_1} - \|\hat{\theta}\|_{\ell_1} \leq \|\hat{\theta}_{\mathcal{J}} - \gamma_{\mathcal{J}}\|_{\ell_1} - \|\hat{\theta}_{\mathcal{J}^c}\|_{\ell_1}.$$

Thus,

$$\begin{aligned}
\|\hat{\theta} - \gamma\|_{\ell_1} &= \|\hat{\theta}_{\mathcal{J}} - \gamma_{\mathcal{J}}\|_{\ell_1} + \|\hat{\theta}_{\mathcal{J}^c}\|_{\ell_1} \\
&\leq 2\|\hat{\theta}_{\mathcal{J}} - \gamma_{\mathcal{J}}\|_{\ell_1} \\
&\leq 2d^{1/2}_{\mathcal{J}}\|\hat{\theta}_{\mathcal{J}} - \gamma_{\mathcal{J}}\|_{\ell_2} \\
&\leq 2d^{1/2}_{\mathcal{J}}\|\hat{\theta} - \gamma\|_{\ell_2} \\
&\leq 2d^{1/2}_{\mathcal{J}}\lambda^{-1}_{\min}(\Sigma^0_{-i,-i})\left[\left(\hat{\theta} - \gamma\right)'\Sigma^0_{-i,-i}\left(\hat{\theta} - \gamma\right)\right]^{1/2} \\
&\leq 2\lambda^{-1}_{\min}(\Sigma_0)d^{1/2}_{\mathcal{J}}\left[\left(\hat{\theta} - \gamma\right)'\Sigma^0_{-i,-i}\left(\hat{\theta} - \gamma\right)\right]^{1/2} \\
&= 2\lambda_{\max}(\Omega_0)d^{1/2}_{\mathcal{J}}\left[\left(\hat{\theta} - \gamma\right)'\Sigma^0_{-i,-i}\left(\hat{\theta} - \gamma\right)\right]^{1/2}.
\end{aligned}$$

Observe that

$$\left(\hat{\theta} - \gamma\right)'\Sigma^0_{-i,-i}\left(\hat{\theta} - \gamma\right) \leq \|\hat{\theta} - \gamma\|_{\ell_1}\left\|\Sigma^0_{-i,-i}\left(\hat{\theta} - \gamma\right)\right\|_{\ell_\infty}.$$

Therefore,

$$\|\hat{\theta} - \gamma\|_{\ell_1} \leq 2\lambda_{\max}(\Omega_0)d^{1/2}_{\mathcal{J}}\|\hat{\theta} - \gamma\|^{1/2}_{\ell_1}\left\|\Sigma^0_{-i,-i}\left(\hat{\theta} - \gamma\right)\right\|^{1/2}_{\ell_\infty},$$

which implies that

$$\|\hat{\theta}-\gamma\|_{\ell_1} \leq 4\lambda_{\max}^2(\Omega_0) d_{\mathcal{G}} \left\|\Sigma^0_{-i,-i}\left(\hat{\theta}-\gamma\right)\right\|_{\ell_\infty}. \tag{12}$$

We now set up to further bound the last term on the right hand side. We appeal to the following result.

Lemma 9 *Under the event that* $\|S-\Sigma_0\|_{\max} < C_0\lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}$, *we have*

$$\left\|\Sigma^0_{-i,-i}\left(\hat{\theta}-\gamma\right)\right\|_{\ell_\infty} \leq 2\delta,$$

provided that

$$\delta \geq \eta\nu + C_0\tau\nu\lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}.$$

Proof By triangular inequality,

$$\left\|\Sigma^0_{-i,-i}\left(\hat{\theta}-\gamma\right)\right\|_{\ell_\infty} \leq \left\|\Sigma^0_{-i,-i}(\theta-\gamma)\right\|_{\ell_\infty} + \left\|\Sigma^0_{-i,-i}\left(\hat{\theta}-\theta\right)\right\|_{\ell_\infty}. \tag{13}$$

We begin with the first term on the right hand side. Recall that

$$\Sigma^0_{-i,i}\Omega^0_{ii} + \Sigma^0_{-i,-i}\Omega^0_{-i,-i} = \mathbf{0},$$

which implies that

$$\Sigma^0_{-i,-i}\theta = \Sigma^0_{-i,i}. \tag{14}$$

Hence,

$$\left\|\Sigma^0_{-i,-i}(\theta-\gamma)\right\|_{\ell_\infty} = \left\|\Sigma^0_{-i,i}-\Sigma^0_{-i,-i}\gamma\right\|_{\ell_\infty} = \Omega_{ii}^{-1}\left\|\Sigma^0_{-i,i}\Omega_{ii}+\Sigma^0_{-i,-i}\Omega_{-i,i}\right\|_{\ell_\infty} \leq \nu\eta.$$

We now turn to the second term on the right hand side of (13). Again by triangular inequality

$$\left\|\Sigma^0_{-i,-i}\left(\hat{\theta}-\theta\right)\right\|_{\ell_\infty} \leq \left\|\left(S_{-i,-i}-\Sigma^0_{-i,-i}\right)\hat{\theta}\right\|_{\ell_\infty} + \left\|S_{-i,-i}\hat{\theta}-\Sigma^0_{-i,-i}\theta\right\|_{\ell_\infty}.$$

To bound the first term on the right hand side, note that

$$\left\|\left(S_{-i,-i}-\Sigma^0_{-i,-i}\right)\hat{\theta}\right\|_{\ell_\infty} \leq \left\|S_{-i,-i}-\Sigma^0_{-i,-i}\right\|_{\max}\|\hat{\theta}\|_{\ell_1} \leq \|S-\Sigma_0\|_{\max}\|\hat{\theta}\|_{\ell_1}.$$

Also recall that

$$\left\|S_{-i,i}-S_{-i,-i}\hat{\theta}\right\|_{\ell_\infty} \leq \delta,$$

and $\Sigma^0_{-i,-i}\theta = \Sigma^0_{-i,i}$. Therefore, by triangular inequality and (14),

$$\left\|S_{-i,-i}\hat{\theta}-\Sigma^0_{-i,-i}\theta\right\|_{\ell_\infty} \leq \delta + \|\Sigma^0_{-i,i}-S_{-i,i}\|_{\ell_\infty} \leq \delta + \|S-\Sigma_0\|_{\max}.$$

To sum up,

$$\begin{aligned}
\left\|\Sigma^0_{-i,-i}\left(\hat{\theta}-\gamma\right)\right\|_{\ell_\infty} &\leq \delta+\nu\eta+\|S-\Sigma_0\|_{\max}+\|S-\Sigma_0\|_{\max}\|\hat{\theta}\|_{\ell_1} \\
&\leq \delta+\nu\eta+\|S-\Sigma_0\|_{\max}(1+\|\gamma\|_{\ell_1}) \\
&\leq \delta+\nu\eta+\|S-\Sigma_0\|_{\max}\|\Omega\|_{\ell_1}/\Omega_{ii} \\
&\leq \delta+\nu\eta+\|S-\Sigma_0\|_{\max}\|\Omega\|_{\ell_1}\lambda_{\min}^{-1}(\Omega) \\
&\leq \delta+\nu\eta+\tau\nu\|S-\Sigma_0\|_{\max},
\end{aligned}$$

which, under the event that $\|S-\Sigma_0\|_{\max} < C_0\lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}$, can be further bounded by 2δ by Lemma 8. ■

Together with (12), Lemma 9 implies that for all $i = 1,\ldots,p$

$$\|\hat{\theta}-\gamma\|_{\ell_1} \le 8\lambda^2_{\max}(\Omega_0)d_{\mathcal{J}}\delta,$$

if $\|S-\Sigma_0\|_{\max} < C_0\lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}$. We are now in position to bound $\|\tilde{\Omega}-\Omega_0\|_{\ell_1}$. We begin with the diagonal elements $|\tilde{\Omega}_{ii}-\Omega^0_{ii}|$.

Lemma 10 *Assume that* $\|S-\Sigma_0\|_{\max} < C_0\lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}$ *and*

$$\delta\lambda_{\max}(\Omega_0)\left(\nu\tau + 8\lambda^2_{\max}(\Omega_0)\lambda^{-1}_{\min}(\Omega_0)d_{\mathcal{J}}\right) + \nu\tau\lambda_{\max}(\Omega_0)\lambda^{-2}_{\min}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1} \le c_0$$

for some numerical constant $0 < c_0 < 1$. *Then*

$$\left|\Omega^0_{ii}-\tilde{\Omega}_{ii}\right| \le \frac{1}{1-c_0}\left(\delta\lambda^2_{\max}(\Omega_0)\left(\nu\tau + 8\lambda^2_{\max}(\Omega_0)d_{\mathcal{J}}\lambda^{-1}_{\min}(\Omega_0)\right) + \nu\tau\lambda^{-2}_{\min}(\Omega_0)\lambda^2_{\max}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1}\right),$$

provided that

$$\delta \ge \eta\nu + C_0\tau\nu\lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}.$$

Proof Recall that $\Sigma^0_{-i,i} = \Sigma^0_{-i,-i}\theta$. Therefore

$$\Omega^0_{ii} = \left(\Sigma^0_{ii} - 2\Sigma^0_{i,-i}\theta + \theta'\Sigma^0_{-i,-i}\theta\right)^{-1} = \left(\Sigma^0_{ii} - \Sigma^0_{i,-i}\theta\right)^{-1}.$$

Because

$$\tilde{\Omega}_{ii} = \left(S_{ii} - 2S_{i,-i}\hat{\theta} + \hat{\theta}'S_{-i,-i}\hat{\theta}\right)^{-1},$$

we have

$$\left|\tilde{\Omega}^{-1}_{ii} - \left(\Omega^0_{ii}\right)^{-1}\right| \le |S_{ii}-\Sigma^0_{ii}| + \left|\hat{\theta}'S_{-i,-i}\hat{\theta} - S_{i,-i}\hat{\theta}\right| + \left|S_{i,-i}\hat{\theta} - \Sigma^0_{i,-i}\theta\right|. \tag{15}$$

We now bound the three terms on the right hand side separately. It is clear that the first term can be bounded by $\|S-\Sigma_0\|_{\max}$. Recall that $\hat{\theta} \in \mathcal{F}$. Hence the second term can be bounded as follows:

$$\left|\hat{\theta}'S_{-i,-i}\hat{\theta} - S_{i,-i}\hat{\theta}\right| \le \left\|S_{-i,-i}\hat{\theta} - S_{-i,i}\right\|_{\ell_\infty}\|\hat{\theta}\|_{\ell_1} \le \delta\|\hat{\theta}\|_{\ell_1}.$$

The last term on the right hand side of (15) can also be bounded similarly.

$$\begin{aligned}
\left|S_{i,-i}\hat{\theta} - \Sigma^0_{i,-i}\theta\right| &\le \left|\left(S_{i,-i}-\Sigma^0_{i,-i}\right)\hat{\theta}\right| + \left|\Sigma^0_{i,-i}\left(\hat{\theta}-\theta\right)\right| \\
&\le \|S-\Sigma_0\|_{\max}\|\hat{\theta}\|_{\ell_1} + \|\Sigma^0_{i,-i}\|_{\ell_\infty}\|\hat{\theta}-\theta\|_{\ell_1} \\
&\le \|S-\Sigma_0\|_{\max}\|\hat{\theta}\|_{\ell_1} + \lambda_{\max}(\Sigma_0)\left(\|\hat{\theta}-\gamma\|_{\ell_1} + \|\gamma-\theta\|_{\ell_1}\right) \\
&= \|S-\Sigma_0\|_{\max}\|\hat{\theta}\|_{\ell_1} + \lambda^{-1}_{\min}(\Omega_0)\left(\|\hat{\theta}-\gamma\|_{\ell_1} + \|\gamma-\theta\|_{\ell_1}\right).
\end{aligned}$$

In summary, we have

$$\begin{aligned}
\left|\tilde{\Omega}^{-1}_{ii} - \left(\Omega^0_{ii}\right)^{-1}\right| &\le \|S-\Sigma_0\|_{\max} + \delta\|\hat{\theta}\|_{\ell_1} + \|S-\Sigma_0\|_{\max}\|\hat{\theta}\|_{\ell_1} \\
&\qquad + \lambda^{-1}_{\min}(\Omega_0)\left(\|\hat{\theta}-\gamma\|_{\ell_1} + \|\gamma-\theta\|_{\ell_1}\right) \\
&\le \nu\tau\|S-\Sigma_0\|_{\max} + \delta\|\hat{\theta}\|_{\ell_1} + \lambda^{-1}_{\min}(\Omega_0)\left(8\lambda^2_{\max}(\Omega_0)d_{\mathcal{J}}\delta + \|\gamma-\theta\|_{\ell_1}\right) \\
&\le \delta\left(\nu\tau + 8\lambda^2_{\max}(\Omega_0)\lambda^{-1}_{\min}(\Omega_0)d_{\mathcal{J}}\right) + \lambda^{-1}_{\min}(\Omega_0)\|\gamma-\theta\|_{\ell_1},
\end{aligned}$$

provided that $\|S-\Sigma_0\|_{\max} < C_0\lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}$. Together with the fact that $\Omega^0_{ii} \leq \lambda_{\max}(\Omega_0)$, this yields

$$\left|\frac{\Omega^0_{ii}}{\tilde{\Omega}_{ii}} - 1\right| \leq \delta\lambda_{\max}(\Omega_0)\left(\nu\tau + 8\lambda^2_{\max}(\Omega_0)\lambda^{-1}_{\min}(\Omega_0)d_{\mathcal{J}}\right) + \lambda_{\max}(\Omega_0)\lambda^{-1}_{\min}(\Omega_0)\|\gamma-\theta\|_{\ell_1}.$$

Moreover, observe that

$$\begin{aligned}\|\gamma-\theta\|_{\ell_1} &\leq \left(\Omega^0_{ii}\right)^{-1}\|\Omega_{-i,i}-\Omega^0_{-i,i}\|_{\ell_1} + \Omega^{-1}_{ii}\left(\Omega^0_{ii}\right)^{-1}|\Omega_{ii}-\Omega^0_{ii}|\|\Omega_{-i,i}\|_{\ell_1} \\ &\leq \lambda^{-1}_{\min}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1} + \lambda^{-1}_{\min}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1}(\nu\tau-1) \\ &\leq \nu\tau\lambda^{-1}_{\min}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1}.\end{aligned}$$

Therefore,

$$\left|\frac{\Omega^0_{ii}}{\tilde{\Omega}_{ii}} - 1\right| \leq \delta\lambda_{\max}(\Omega_0)\left(\nu\tau + 8\lambda^2_{\max}(\Omega_0)\lambda^{-1}_{\min}(\Omega_0)d_{\mathcal{J}}\right) + \nu\tau\lambda_{\max}(\Omega_0)\lambda^{-2}_{\min}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1}, \quad (16)$$

which implies that

$$\frac{\Omega^0_{ii}}{\tilde{\Omega}_{ii}} \geq 1 - c_0.$$

Subsequently,

$$\tilde{\Omega}_{ii} \leq \frac{1}{1-c_0}\Omega^0_{ii} \leq \frac{1}{1-c_0}\lambda_{\max}(\Omega_0).$$

Together with (16), this implies

$$\begin{aligned}\left|\Omega^0_{ii}-\tilde{\Omega}_{ii}\right| &\leq \tilde{\Omega}_{ii}\left|\frac{\Omega^0_{ii}}{\tilde{\Omega}_{ii}} - 1\right| \\ &\leq \frac{1}{1-c_0}\delta\lambda^2_{\max}(\Omega_0)\left(\nu\tau + 8\lambda^2_{\max}(\Omega_0)\lambda^{-1}_{\min}(\Omega_0)d_{\mathcal{J}}\right) \\ &\qquad + \frac{1}{1-c_0}\nu\tau\lambda^{-2}_{\min}(\Omega_0)\lambda^2_{\max}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1}.\end{aligned}$$

■

We now turn to the off-diagonal entries of $\tilde{\Omega}-\Omega_0$.

Lemma 11 *Under the assumptions of Lemma 10, there exist positive constants C_1, C_2 and C_3 depending only on ν, τ, $\lambda_{\min}(\Omega_0)$ and $\lambda_{\max}(\Omega_0)$ such that*

$$\|\tilde{\Omega}_{-i,\cdot} - \Omega^0_{-i,\cdot}\|_{\ell_1} \leq (C_1 + C_2 d_{\mathcal{J}})\delta + C_3\|\Omega-\Omega_0\|_{\ell_1}.$$

Proof Note that

$$\begin{aligned}
\|\tilde{\Omega}_{-i,i}-\Omega^0_{-i,i}\|_{\ell_1} &= \left\|\tilde{\Omega}_{ii}\hat{\theta}-\Omega^0_{ii}\theta\right\|_{\ell_1} \\
&\le \Omega^0_{ii}\|\hat{\theta}-\theta\|_{\ell_1}+\left|\tilde{\Omega}_{ii}-\Omega^0_{ii}\right|\|\hat{\theta}\|_{\ell_1} \\
&\le \lambda^{-1}_{\min}(\Omega_0)\left(\|\hat{\theta}-\gamma\|_{\ell_1}+\|\gamma-\theta\|_{\ell_1}\right) \\
&\qquad +\frac{\nu\tau-1}{1-c_0}\delta\lambda^2_{\max}(\Omega_0)\left(\nu\tau+8\lambda^2_{\max}(\Omega_0)d_J\lambda^{-1}_{\min}(\Omega_0)\right) \\
&\qquad +\frac{\nu\tau-1}{1-c_0}\nu\tau\lambda^{-2}_{\min}(\Omega_0)\lambda^2_{\max}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1} \\
&\le 8\nu^2 d_J\lambda^{-1}_{\min}(\Omega_0)\delta+\nu\tau\lambda^{-2}_{\min}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1} \\
&\qquad +\frac{\nu\tau-1}{1-c_0}\delta\lambda^2_{\max}(\Omega_0)\left(\nu\tau+8\lambda^2_{\max}(\Omega_0)d_J\lambda^{-1}_{\min}(\Omega_0)\right) \\
&\qquad +\frac{\nu\tau-1}{1-c_0}\nu\tau\lambda^{-2}_{\min}(\Omega_0)\lambda^2_{\max}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\|\tilde{\Omega}_{-i,\cdot}-\Omega^0_{-i,\cdot}\|_{\ell_1} &= \left|\Omega^0_{ii}-\tilde{\Omega}_{ii}\right|+\|\tilde{\Omega}_{-i,i}-\Omega^0_{-i,i}\|_{\ell_1} \\
&\le \delta\left(\frac{1}{1-c_0}\nu^2\tau^2\lambda^2_{\max}(\Omega_0)+8\left(1+\frac{\nu\tau}{1-c_0}\right)\lambda^2_{\max}(\Omega_0)d_J\lambda^{-1}_{\min}(\Omega_0)\right) \\
&\qquad +\left(1+\frac{\nu\tau}{1-c_0}\lambda^2_{\max}(\Omega_0)\right)\nu\tau\lambda^{-2}_{\min}(\Omega_0)\|\Omega-\Omega_0\|_{\ell_1}.
\end{aligned}$$

■

From Lemma 11, it is clear that under the assumptions of Lemma 10,

$$\|\tilde{\Omega}-\Omega^0\|_{\ell_1} \le C\inf_{\Omega\in O(\nu,\eta,\tau)}\left(\|\Omega-\Omega_0\|_{\ell_1}+\deg(\Omega)\delta\right), \tag{17}$$

where $C=\max\{C_1,C_2,C_3\}$ is a positive constant depending only on ν,τ, $\lambda_{\min}(\Omega_0)$ and $\lambda_{\max}(\Omega_0)$. By the definition of $\hat{\Omega}$,

$$\|\hat{\Omega}-\tilde{\Omega}\|_{\ell_1} \le \|\tilde{\Omega}-\Omega_0\|_{\ell_1} \le C\inf_{\Omega\in O(\nu,\eta,\tau)}\left(\|\Omega-\Omega_0\|_{\ell_1}+\deg(\Omega)\delta\right).$$

An application of triangular inequality immediately gives

$$\begin{aligned}
\|\hat{\Omega}-\Omega_0\|_{\ell_1} &\le \|\hat{\Omega}-\tilde{\Omega}\|_{\ell_1}+\|\tilde{\Omega}-\Omega_0\|_{\ell_1} \\
&\le 2C\inf_{\Omega\in O(\nu,\eta,\tau)}\left(\|\Omega-\Omega_0\|_{\ell_1}+\deg(\Omega)\delta\right).
\end{aligned}$$

To complete the proof, we appeal to the following lemma showing that

$$\|S-\Sigma_0\|_{\max} \le C_0\lambda_{\max}(\Sigma_0)\sqrt{\frac{t+\log p}{n}},$$

for a numerical constant $C_0>0$, with probability at least $1-e^{-t}$. Taking $t=A\log p$ yields

$$\mathbb{P}\left\{\|S-\Sigma_0\|_{\max} < C_0\lambda_{\max}(\Sigma_0)((A+1)n^{-1}\log p)^{1/2}\right\} \ge 1-p^{-A}.$$

Lemma 12 *Assume that there exist constants $c_0 \geq 0$, and $T > 0$ such that for any $|t| \leq T$*

$$\mathbb{E}e^{tX_i^2} \leq c_0, \qquad i = 1, 2, \ldots, p.$$

Then there exists a constant $C > 0$ depending on c_0 and T such that

$$\|S - \Sigma_0\|_{\max} \leq C\sqrt{\frac{t + \log p}{n}}$$

with probability at least $1 - e^{-t}$ for all $t > 0$.

Proof Observe that S is invariant to $\mathbb{E}X$. We shall assume that $\mathbb{E}X = \mathbf{0}$ without loss of generality. Note that $S_{ij} = \mathbb{E}_n X_i X_j - \mathbb{E}_n X_i \mathbb{E}_n X_j$. We have

$$|S_{ij} - \Sigma^0_{ij}| \leq |\mathbb{E}_n X_i X_j - \mathbb{E}X_i X_j| + |\mathbb{E}_n X_i||\mathbb{E}_n X_j| =: \Delta_1 + \Delta_2.$$

We begin by bounding Δ_1.

$$\begin{aligned} \left|(\mathbb{E}_n - \mathbb{E})X_i X_j\right| &= \frac{1}{4}\left|(\mathbb{E}_n - \mathbb{E})\left((X_i + X_j)^2 - (X_i - X_j)^2\right)\right| \\ &\leq \frac{1}{4}\left|(\mathbb{E}_n - \mathbb{E})(X_i + X_j)^2\right| + \frac{1}{4}\left|(\mathbb{E}_n - \mathbb{E})(X_i - X_j)^2\right|. \end{aligned}$$

The two terms in the upper bound can be bounded similarly and we focus only on the first term. By the sub-Gaussianity of X_i and X_j, for any $|t| \leq T/4$,

$$\mathbb{E}e^{t(X_i + X_j)^2} \leq \mathbb{E}e^{2tX_i^2} e^{2tX_j^2} \leq \mathbb{E}^{1/2} e^{4tX_i^2} \mathbb{E}^{1/2} e^{4tX_j^2} \leq c_0.$$

In other words, $\{X_i + X_j : 1 \leq i, j \leq p\}$ are also sub-Gaussian. Observe that

$$\begin{aligned} \ln\left(\mathbb{E}e^{t[(X_i + X_j)^2 - \mathbb{E}(X_i + X_j)^2]}\right) &= \ln \mathbb{E}e^{t(X_i + X_j)^2} - t\mathbb{E}(X_i + X_j)^2 \\ &\leq \mathbb{E}\left[e^{t(X_i + X_j)^2} - t(X_i + X_j)^2 - 1\right], \end{aligned}$$

where we used the fact that $\ln u \leq u - 1$ for all $u > 0$. An application of the Taylor expansion now yields that there exist constants $c_1, T_1 > 0$ such that

$$\ln\left(\mathbb{E}e^{t[(X_i + X_j)^2 - \mathbb{E}(X_i + X_j)^2]}\right) \leq c_1 t^2$$

for all $|t| < T_1$. In other words,

$$\mathbb{E}e^{t[(X_i + X_j)^2 - \mathbb{E}(X_i + X_j)^2]} \leq e^{c_1 t^2}$$

for any $|t| < T_1$. Therefore,

$$\mathbb{E}e^{t[(\mathbb{E}_n - \mathbb{E})(X_i + X_j)^2]} \leq e^{c_1 t^2/n}.$$

By Markov inequality

$$\mathbb{P}\left\{(\mathbb{E}_n - \mathbb{E})(X_i + X_j)^2 \geq x\right\} \leq e^{-tx}\mathbb{E}e^{t[(\mathbb{E}_n - \mathbb{E})(X_i + X_j)^2]} \leq \exp\left(c_1 t^2/n - tx\right).$$

Taking $t = nx/2c_1$ yields

$$\mathbb{P}\left\{(\mathbb{E}_n - \mathbb{E})(X_i + X_j)^2 \geq x\right\} \leq \exp\left\{-\frac{nx^2}{4c_1}\right\}.$$

Similarly,

$$\mathbb{P}\left\{(\mathbb{E}_n - \mathbb{E})(X_i + X_j)^2 \leq -x\right\} \leq \exp\left\{-\frac{nx^2}{4c_1}\right\}.$$

Therefore,

$$\mathbb{P}\left\{\left|(\mathbb{E}_n - \mathbb{E})(X_i + X_j)^2\right| \geq x\right\} \leq 2\exp\left\{-\frac{nx^2}{4c_1}\right\}.$$

Following the same argument, one can show that

$$\mathbb{P}\left\{\left|(\mathbb{E}_n - \mathbb{E})(X_i - X_j)^2\right| \geq x\right\} \leq 2\exp\left\{-\frac{nx^2}{4c_1}\right\}.$$

Note also that this inequality holds trivially when $i = j$. In summary, we have

$$\begin{aligned}\mathbb{P}\{\Delta_1 \geq x\} &\leq \mathbb{P}\left\{\left|(\mathbb{E}_n - \mathbb{E})(X_i + X_j)^2\right| \geq 2x\right\} + \mathbb{P}\left\{\left|(\mathbb{E}_n - \mathbb{E})(X_i - X_j)^2\right| \geq 2x\right\} \\ &\leq 4\exp\left\{-c_1^{-1}nx^2\right\}.\end{aligned}$$

Now consider Δ_2.

$$\mathbb{E}e^{t\mathbb{E}_n X_i \mathbb{E}_n X_j} \leq \mathbb{E}^{1/2} e^{2t\mathbb{E}_n X_i} \mathbb{E}^{1/2} e^{2t\mathbb{E}_n X_j} \leq \max_{1\leq i\leq p} \mathbb{E}e^{2t\mathbb{E}_n X_i}.$$

Following a similar argument as before, we can show that there exist constants $c_2, T_2 > 0$ such that

$$\mathbb{E}e^{2t\mathbb{E}_n X_i} \leq e^{c_2 t^2/n}$$

for all $|t| < T_2$. This further leads to, similar to before,

$$\mathbb{P}\{\Delta_1 \geq x\} \leq 2\exp\left\{-c_2^{-1}nx^2\right\}.$$

To sum up,

$$\begin{aligned}\mathbb{P}\left\{\|S - \Sigma_0\|_{\max} \geq x\right\} &\leq p^2 \max_{1\leq i,j\leq p} \mathbb{P}\left\{|S_{ij} - \Sigma^0_{ij}\| \geq x\right\} \\ &\leq p^2\left[\mathbb{P}\left(\Delta_1 \geq x/2\right) + \mathbb{P}\left(\Delta_2 \geq x/2\right)\right] \\ &\leq 4p^2\exp\left\{-c_3 nx^2\right\}.\end{aligned}$$

for some constant $c_3 > 0$. The claimed result now follows. ■

6.2 Proof of Theorem 5

First note that the claim follows from

$$\inf_{\bar{\Omega}} \sup_{\Omega_0 \in \mathcal{M}_1(\tau_0,\nu_0,d)} \mathbb{E}\left\|\bar{\Omega}-\Omega_0\right\|_{\ell_1} \geq C' d\sqrt{\frac{\log p}{n}} \tag{18}$$

for some constant $C' > 0$. We establish the minimax lower bound (18) using the tools from Korostelev and Tsybakov (1993), which is based upon testing many hypotheses as well as statistical applications of Fano's lemma and the Varshamov-Gilbert bound. More specifically, it suffices to find a collection of inverse covariance matrices $\mathcal{M}' = \{\Omega_1, \ldots, \Omega_{K+1}\} \subset \mathcal{M}_1(\tau_0, \nu_0, d)$ such that

(a) for any two distinct members $\Omega_j, \Omega_k \in \mathcal{M}'$, $\|\Omega_j - \Omega_k\|_{\ell_1} > Ad(n^{-1}\log p)^{1/2}$ for some constant $A > 0$;

(b) there exists a numerical constant $0 < c_0 < 1/8$ such that

$$\frac{1}{K}\sum_{k=1}^{K} \mathcal{K}(\mathcal{P}(\Omega_k), \mathcal{P}(\Omega_{K+1})) \leq c_0 \log K,$$

where $\mathcal{K}$ stands for the Kullback-Leibler divergence and $\mathcal{P}(\Omega)$ is the probability measure $\mathcal{N}(\mathbf{0}, \Omega)$.

To construct $\mathcal{M}'$, we assume that $d(n^{-1}\log p)^{1/2} < 1/2$, $\tau_0, \nu_0 > 2$ without loss of generality. As shown by Birgé and Massart (1998), from the Varshamov-Gilbert bound, there is a set of binary vectors of length $p-1$, $\mathcal{B} = \{b_1, \ldots, b_K\} \subset \{0,1\}^{p-1}$ such that (i) there are d ones in a vector b_j for any $j = 1, \ldots, p-1$; (ii) the Hamming distance between b_j and b_k is at least $d/2$ for any $j \neq k$; (iii) $\log K > 0.233 d \log(p/d)$. We now take Ω_k for $k = 1, \ldots, K$ as follows. It differs from the identity matrix only by its first row and column. More specifically, $\Omega^k_{11} = 1$, $\Omega^k_{-1,1} = (\Omega^k_{1,-1})' = a_n b_k$, $\Omega^k_{-1,-1} = I_{p-1}$, that is,

$$\Omega_k = \begin{pmatrix} 1 & a_n b_k' \\ a_n b_k & I \end{pmatrix},$$

where $a_n = a_0(n^{-1}\log p)^{1/2}$ with a constant $0 < a_0 < 1$ to be determined later. Finally, we take $\Omega_{K+1} = I$. It is clear that Condition (a) is satisfied with this choice of $\mathcal{M}'$ and $A = a_0/2$. It remains to verify Condition (b). Simple algebraic manipulations yield that for any $1 \leq k \leq K$,

$$\mathcal{K}(\mathcal{P}(\Omega_k), \mathcal{P}(\Omega_{K+1})) = -\frac{n}{2}\log\det(\Omega_k) = -\frac{n}{2}\log(1 - a_n^2 b_k' b_k).$$

Recall that $a_n^2 b_k' b_k = d a_n^2 < d^2 a_n^2 < 1/4$. Together with the fact that $-\log(1-x) \leq \log(1+2x) \leq 2x$ for $0 < x < 1/2$, we have

$$\mathcal{K}(\mathcal{P}(\Omega_k), \mathcal{P}(\Omega_{K+1})) \leq n d a_n^2.$$

By setting a_0 small enough, this can be further bounded by $0.233 c_0 d \log(p/d)$ and subsequently $c_0 \log K$. The proof is now completed. ∎

6.3 Proof of Theorem 6

We prove the theorem by applying the oracle inequality from Theorem 1. To this end, we need to find an "oracle" inverse covariance matrix Ω. Let

$$\Omega_{ij} = \Omega^0_{ij} \mathbf{1}\left(\left|\Omega^0_{ij}\right| \geq \zeta\right),$$

where $\zeta > 0$ is to be specified later. We now verify that $\Omega \in O(\nu, \eta, \tau)$ with appropriate choices of the three parameters.

First observe that

$$\begin{aligned}
\|\Omega - \Omega_0\|_{\ell_1} &\leq \max_{1\leq i\leq p} \sum_{j=1}^{p} \left|\Omega^0_{ij}\right| \mathbf{1}\left(\left|\Omega^0_{ij}\right| \leq \zeta\right) \\
&\leq \zeta^{1-\alpha} \max_{1\leq i\leq p} \sum_{j=1}^{p} \left|\Omega^0_{ij}\right|^{\alpha} \mathbf{1}\left(\left|\Omega^0_{ij}\right| \leq \zeta\right) \\
&\leq M\zeta^{1-\alpha}.
\end{aligned}$$

Thus,

$$\nu_0^{-1} - M\zeta^{1-\alpha} \leq \lambda_{\min}(\Omega) \leq \lambda_{\max}(\Omega) \leq \nu_0 + M\zeta^{1-\alpha}.$$

In particular, setting ζ small enough such that $M\zeta^{1-\alpha} < (2\nu_0)^{-1}$ yields

$$(2\nu_0)^{-1} \leq \lambda_{\min}(\Omega) \leq \lambda_{\max}(\Omega) \leq 2\nu_0.$$

We can therefore take $\nu = 2\nu_0$.

Now consider the approximation error $\|\Sigma_0\Omega - I\|_{\max}$. Note that the (i,j) entry of $\Sigma_0\Omega - I = \Sigma_0(\Omega - \Omega_0)$ can be bounded as follows

$$\begin{aligned}
\left|\sum_{k=1}^{p} \Sigma^0_{ik}\Omega^0_{kj} \mathbf{1}\left(\left|\Omega^0_{kj}\right| \leq \zeta\right)\right| &\leq \sum_{k=1}^{p} \left|\Sigma^0_{ik}\right| \left|\Omega^0_{kj}\right| \mathbf{1}\left(\left|\Omega^0_{kj}\right| \leq \zeta\right) \\
&\leq \zeta \max_{1\leq k\leq p} \sum_{k=1}^{p} \left|\Sigma^0_{ik}\right| \\
&= \zeta\|\Sigma_0\|_{\ell_1}.
\end{aligned}$$

This implies that

$$\|\Sigma_0\Omega - I\|_{\max} \leq \zeta\|\Sigma_0\|_{\ell_1}.$$

In other words, we can take $\eta = \zeta\|\Sigma_0\|_{\ell_1}$.

Furthermore, it is clear that we can take $\|\Omega\|_{\ell_1} \leq \|\Omega_0\|_{\ell_1}$. Therefore, by Theorem 1, there exist constants $C_1, C_2 > 0$ depending only on $\|\Omega_0\|_{\ell_1}$, $\|\Sigma_0\|_{\ell_1}$, and ν_0 such that for any

$$\delta \geq C_1 \left(\zeta + \sqrt{\frac{(A+1)\log p}{n}}\right) \qquad (i = 1, 2, \ldots, p),$$

we have

$$\|\hat{\Omega} - \Omega_0\|_{\ell_1} \leq C_2\left(M\zeta^{1-\alpha} + \deg(\Omega)\delta\right)$$

with probability at least $1-p^{-A}$. Now note that

$$\deg(\Omega) = \max_{1\le i\le p} \sum_{j=1}^{p} \mathbf{1}\left(\left|\Omega_{ij}^0\right| \ge \zeta\right) \le M\zeta^{-\alpha}.$$

The claimed the results then follows by taking $\zeta = C_3((A+1)n^{-1}\log p)^{1/2}$ for a small enough constant $C_3 > 0$ such that $M\zeta^{1-\alpha} < (2\nu_0)^{-1}$.

6.4 Proof of Theorem 7

Assume that $M(n^{-1}\log p)^{(1-\alpha)/2} < 1/2, \tau_0, \nu_0 > 2$ without loss of generality. Similar to Theorem 5, there eixs a collection of inverse covariance matrices $\mathcal{M}' = \{\Omega_1, \ldots, \Omega_{K+1}\} \subset \mathcal{M}_2$ such that

(a) for any two distinct members $\Omega_j, \Omega_k \in \mathcal{M}'$, $\|\Omega_j - \Omega_k\|_{\ell_1} > AM(n^{-1}\log p)^{(1-\alpha)/2}$ for some constant $A > 0$;

(b) there exists a numerical constant $0 < c_0 < 1/8$ such that

$$\frac{1}{K}\sum_{k=1}^{K} \mathcal{K}(\mathcal{P}(\Omega_k), \mathcal{P}(\Omega_{K+1})) \le c_0 \log K.$$

To this end, we follow the same construction as in the proof of Theorem 5 by taking

$$d = \left\lfloor M\left(\frac{\log p}{n}\right)^{-\frac{\alpha}{2}} \right\rfloor,$$

and $\lfloor x \rfloor$ stands for the integer part of x. First, we need to show that $\mathcal{M}' \subset \mathcal{M}_2$. Because $d(n^{-1}\log p)^{1/2} < 1/2$, it is clear that the bounded eigenvalue condition and $\|\Omega_k\|_{\ell_1} < \tau_0$ can ensured by setting a_0 small enough. It is also obvious that

$$\max_{1\le j\le p} \sum_{i=1}^{p} |\Omega_{ij}^k|^{\alpha} \le M.$$

It remains to check that $\|\Sigma_k\|_{\ell_1}$ is bounded. By the block matrix inversion formula

$$\Sigma_k = \begin{pmatrix} \frac{1}{1-a_n^2 b_k' b_k} & -\frac{a_n}{1-a_n^2 b_k' b_k} b_k' \\ -\frac{a_n}{1-a_n^2 b_k' b_k} b_k & I + \frac{a_n^2}{1-a_n^2 b_k' b_k} b_k b_k' \end{pmatrix}.$$

It can then be readily checked that $\|\Sigma_k\|_{\ell_1}$ can also be bounded from above by setting a_0 small enough.

Next we verify Conditions (a) and (b). It is clear that Condition (a) is satisfied with this choice of $\mathcal{M}'$ and $A = a_0/2$. It remains to verify Condition (b). Simple algebraic manipulations yield that for any $1 \le k \le K$,

$$\mathcal{K}(\mathcal{P}(\Omega_k), \mathcal{P}(\Omega_{K+1})) = -\frac{n}{2}\log\det(\Omega_k) = -\frac{n}{2}\log(1 - a_n^2 b_k' b_k).$$

Recall that $a_n^2 b_k' b_k = d a_n^2 < d^2 a_n^2 < 1/4$. Together with the fact that $-\log(1-x) \le \log(1+2x) \le 2x$ for $0 < x < 1/2$, we have

$$\mathcal{K}(\mathcal{P}(\Omega_k), \mathcal{P}(\Omega_{K+1})) \le n d a_n^2.$$

By setting a_0 small enough, this can be further bounded by $0.233c_0d\log(p/d)$ and subsequently $c_0\log K$. The proof of (9) is now completed.

The proof of (10) follows from a similar argument. We essentially construct the same subset $\mathcal{M}'$ but with

$$\Sigma_k = \begin{pmatrix} 1 & a_n b_k' \\ a_n b_k & I \end{pmatrix}.$$

The only difference from before is the calculation of Kullback-Leibler divergence, which in this case is

$$\mathcal{K}(\mathcal{P}(\Sigma_k), \mathcal{P}(\Sigma_{K+1})) = \frac{n}{2}\left(\text{trace}(\Omega_k) + \log\det(\Sigma_k) - p\right)$$

where

$$\Omega_k = \begin{pmatrix} \frac{1}{1-a_n^2 b_k' b_k} & -\frac{a_n}{1-a_n^2 b_k' b_k} b_k' \\ -\frac{a_n}{1-a_n^2 b_k' b_k} b_k & I + \frac{a_n^2}{1-a_n^2 b_k' b_k} b_k b_k' \end{pmatrix}.$$

Therefore, $\text{trace}(\Omega_k) = p + 2da_n^2/(1-da_n^2)$. Together with the fact that $\det(\Sigma_k) = 1 - da_n^2$, we conclude that

$$\mathcal{K}(\mathcal{P}(\Sigma_k), \mathcal{P}(\Sigma_{K+1})) = \frac{n}{2}\left(\frac{2da_n^2}{(1-da_n^2)} + \log(1-da_n^2)\right) \leq \frac{1}{2} n d a_n^2.$$

where we used the fact that $\log(1-x) \leq -x$. The rest of the argument proceeds in the same fashion as before.

Acknowledgments

This was supported in part by NSF grant DMS-0846234 (CAREER) and a grant from Georgia Cancer Coalition. The author wish to thank the editor and three anonymous referees for their comments that help greatly improve the manuscript.

References

T.W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley-Interscience, London, 2003.

M. Asif. *Primal dual pursuit: a homotopy based algorithm for the Dantzig selector*. Master Thesis, School of Electrical and Computer Enginnering, Georgia Institute of Technology, 2008.

O. Banerjee, L. El Ghaoui and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485-516, 2008.

P. Bickel and E. Levina. Regularized estimation of large covariance matrices. *Annals of Statistics*, 36:199-227, 2008a.

P. Bickel and E. Levina. Covariance regularization by thresholding. *Annals of Statistics*, 36:2577-2604, 2008b.

P. Bickel, Y. Ritov and A. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *Annals of Statistics*, 37:1705-1732, 2009.

L. Birgé and P. Massart. Minimum contrast estimators on sieves: exponential bounds and rates of convergence. *Bernoulli*, 4(3):329-375, 1998.

L. Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37:373-384, 1995.

T.T. Cai, C. Zhang, and H. Zhou. Optimal rates of convergence for covariance matrix estimation. *Annals of Statistics*, 38:2118-2144, 2010.

E.J. Candés and T. Tao. The Dantzig selector: statistical estimation when p is much larger than n. *Annals of Statistics*, 35:2313-2351, 2007.

A. d'Aspremont, O. Banerjee and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and it Applications*, 30:56-66, 2008.

A. Dempster. Covariance selection. *Biometrika*, 32:95-108, 1972.

X. Deng and M. Yuan. Large Gaussian covariance matrix estimation with Markov structures. *Journal of Computational and Graphical Statistics*, 18:640-657, 2008.

D.M. Edwards. *Introduction to Graphical Modelling*, Springer, New York, 2000.

B. Efron, T. Hastie, I. Johnstone and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407-499, 2004.

N. El Karoui. Operator norm consistent estimation of large dimensional sparse covariance matrices. *Annals of Statistics*, 36:2717-2756, 2008.

J. Fan, Y. Fan and J. Lv. High dimensional covariance matrix estimation using a factor model. *Journal of Econometrics*, 147:186-197, 2008.

J. Friedman, T. Hastie and T. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432-441, 2008.

J. Huang, N. Liu, M. Pourahmadi and L. Liu. Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93:85-98, 2006.

A. Korostelev and A. Tsybakov. *Minimax Theory of Image Reconstruction*. Springer, New York, 1993.

C. Lam and J. Fan. Sparsistency and rates of convergence in large covariance matrices estimation. *Annals of Statistics*, 37:4254-4278, 2009.

S.L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365-411, 2004.

E. Levina, A.J. Rothman and J. Zhu. Sparse estimation of large covariance matrices via a nested lasso penalty. *Annals of Applied Statistics*, 2:245-263, 2007.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34:1436-1462, 2006.

R. Muirhead. *Aspects of Multivariate Statistical Theory*. Wiley, London, 2005.

M. Pourahmadi. Joint mean-covariance models with applications to longitudinal data: unconstrained parameterisation. *Biometrika*, 86:677-690, 1999.

M. Pourahmadi. Maximum likelihood estimation of generalized linear models for multivariate normal covariance matrix. *Biometrika*, 87:425-435, 2000.

P. Ravikumar, G. Raskutti, M. Wainwright and B. Yu. Model selection in Gaussian graphical models: high-dimensional consistency of ℓ_1-regularized MLE. In *Advances in Neural Information Processing Systems (NIPS)* 21, 2008.

P. Ravikumar, M. Wainwright, G. Raskutti and B. Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *Technical Report*, 2008.

G. Rocha, P. Zhao and B. Yu. A path following algorithm for sparse pseudo-likelihood inverse covariance estimation. *Technical Report*, 2008.

A. Rothman, P. Bickel, E. Levina and J. Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494-515, 2008.

A. Rothman, E. Levina and J. Zhu. Generalized thresholding of large covariance matrices. *Journal of the American Statistical Association*, 104:177-186, 2009.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267-288, 1996.

J. Whittaker. *Graphical Models in Applied Multivariate Statistics*, John Wiley and Sons, Chichester, 1990.

W. Wu and M. Pourahmadi. Nonparametric estimation of large covariance matrices of longitudinal data. *Biometrika*, 90:831-844, 2003.

M. Yuan. Efficient computation of the ℓ_1 regularized solution path in Gaussian graphical models. *Journal of Computational and Graphical Statistics*, 17:809-826, 2008.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94:19-35, 2007.

Journal of Machine Learning Research 11 (2010) 2287-2322 Submitted 7/09; Revised 4/10; Published 8/10

Spectral Regularization Algorithms for Learning Large Incomplete Matrices

Rahul Mazumder RAHULM@STANFORD.EDU
Trevor Hastie* HASTIE@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305

Robert Tibshirani† TIBS@STANFORD.EDU
Department of Health, Research and Policy
Stanford University
Stanford, CA 94305

Editor: Tommi Jaakkola

Abstract

We use convex relaxation techniques to provide a sequence of regularized low-rank solutions for large-scale matrix completion problems. Using the nuclear norm as a regularizer, we provide a simple and very efficient convex algorithm for minimizing the reconstruction error subject to a bound on the nuclear norm. Our algorithm SOFT-IMPUTE iteratively replaces the missing elements with those obtained from a soft-thresholded SVD. With warm starts this allows us to efficiently compute an entire regularization path of solutions on a grid of values of the regularization parameter. The computationally intensive part of our algorithm is in computing a low-rank SVD of a dense matrix. Exploiting the problem structure, we show that the task can be performed with a complexity of order linear in the matrix dimensions. Our semidefinite-programming algorithm is readily scalable to large matrices; for example SOFT-IMPUTE takes a few hours to compute low-rank approximations of a $10^6 \times 10^6$ incomplete matrix with 10^7 observed entries, and fits a rank-95 approximation to the full Netflix training set in 3.3 hours. Our methods achieve good training and test errors and exhibit superior timings when compared to other competitive state-of-the-art techniques.

Keywords: collaborative filtering, nuclear norm, spectral regularization, netflix prize, large scale convex optimization

1. Introduction

In many applications measured data can be represented in a matrix $X_{m\times n}$, for which only a relatively small number of entries are observed. The problem is to "complete" the matrix based on the observed entries, and has been dubbed the matrix completion problem (Candès and Recht, 2008; Candès and Tao, 2009; Rennie and Srebro, 2005). The "Netflix" competition (for example, SIGKDD and Netflix, 2007) is a popular example, where the data is the basis for a recommender system. The rows correspond to viewers and the columns to movies, with the entry X_{ij} being the rating $\in \{1,\ldots,5\}$ by viewer i for movie j. There are about 480K viewers and 18K movies, and hence 8.6 billion (8.6×10^9) potential entries. However, on average each viewer rates about 200

*. Also in the Department of Health, Research and Policy.
†. Also in the Department of Statistics.

movies, so only 1.2% or 10^8 entries are observed. The task is to predict the ratings that viewers would give to movies they have not yet rated.

These problems can be phrased as learning an unknown parameter (a matrix $Z_{m\times n}$) with very high dimensionality, based on very few observations. In order for such inference to be meaningful, we assume that the parameter Z lies in a much lower dimensional manifold. In this paper, as is relevant in many real life applications, we assume that Z can be well represented by a matrix of low rank, that is, $Z \approx V_{m\times k}G_{k\times n}$, where $k \ll \min(n,m)$. In this recommender-system example, low rank structure suggests that movies can be grouped into a small number of "genres", with $G_{\ell j}$ the relative score for movie j in genre ℓ. Viewer i on the other hand has an affinity $V_{i\ell}$ for genre ℓ, and hence the modeled score for viewer i on movie j is the sum $\sum_{\ell=1}^{k} V_{i\ell}G_{\ell j}$ of genre affinities times genre scores. Typically we view the observed entries in X as the corresponding entries from Z contaminated with noise.

Srebro et al. (2005a) studied generalization error bounds for learning low-rank matrices. Recently Candès and Recht (2008), Candès and Tao (2009), and Keshavan et al. (2009) showed theoretically that under certain assumptions on the entries of the matrix, locations, and proportion of unobserved entries, the true underlying matrix can be recovered within very high accuracy.

For a matrix $X_{m\times n}$ let $\Omega \subset \{1,\ldots,m\}\times\{1,\ldots,n\}$ denote the indices of observed entries. We consider the following optimization problem:

$$\begin{aligned} &\text{minimize} && \operatorname{rank}(Z) \\ &\text{subject to} && \sum_{(i,j)\in\Omega} (X_{ij}-Z_{ij})^2 \le \delta, \end{aligned} \tag{1}$$

where $\delta \ge 0$ is a regularization parameter controlling the tolerance in training error. The rank constraint in (1) makes the problem for general Ω combinatorially hard (Srebro and Jaakkola, 2003). For a fully-observed X on the other hand, the solution is given by a truncated singular value decomposition (SVD) of X. The following seemingly small modification to (1),

$$\begin{aligned} &\text{minimize} && \|Z\|_* \\ &\text{subject to} && \sum_{(i,j)\in\Omega} (X_{ij}-Z_{ij})^2 \le \delta, \end{aligned} \tag{2}$$

makes the problem convex (Fazel, 2002). Here $\|Z\|_*$ is the nuclear norm, or the sum of the singular values of Z. Under many situations the nuclear norm is an effective convex relaxation to the rank constraint (Fazel, 2002; Candès and Recht, 2008; Candès and Tao, 2009; Recht et al., 2007). Optimization of (2) is a semi-definite programming problem (Boyd and Vandenberghe, 2004) and can be solved efficiently for small problems, using modern convex optimization software like SeDuMi and SDPT3 (Grant and Boyd., 2009). However, since these algorithms are based on second order methods (Liu and Vandenberghe, 2009), they can become prohibitively expensive if the dimensions of the matrix get large (Cai et al., 2008). Equivalently we can reformulate (2) in *Lagrange* form

$$\underset{Z}{\text{minimize}}\, \frac{1}{2}\sum_{(i,j)\in\Omega} (X_{ij}-Z_{ij})^2 + \lambda\|Z\|_*. \tag{3}$$

Here $\lambda \ge 0$ is a regularization parameter controlling the nuclear norm of the minimizer $\hat{Z}_\lambda$ of (3); there is a 1-1 mapping between $\delta \ge 0$ and $\lambda \ge 0$ over their active domains.

In this paper we propose an algorithm SOFT-IMPUTE for the nuclear norm regularized least-squares problem (3) that scales to large problems with $m,n \approx 10^5$–10^6 with around 10^6–10^8 or more observed entries. At every iteration SOFT-IMPUTE decreases the value of the objective function towards its minimum, and at the same time gets closer to the set of optimal solutions of the problem (2). We study the convergence properties of this algorithm and discuss how it can be extended to other more sophisticated forms of spectral regularization.

To summarize some performance results[1]

- We obtain a rank-40 solution to (2) for a problem of size $10^5 \times 10^5$ and $|\Omega| = 5 \times 10^6$ observed entries in less than 18 minutes.
- For the same sized matrix with $|\Omega| = 10^7$ we obtain a rank-5 solution in less than 21 minutes.
- For a $10^6 \times 10^5$ sized matrix with $|\Omega| = 10^8$ a rank-5 solution is obtained in approximately 4.3 hours.
- We fit a rank-66 solution for the Netflix data in 2.2 hours. Here there are 10^8 observed entries in a matrix with 4.8×10^5 rows and 1.8×10^4 columns. A rank 95 solution takes 3.27 hours.

The paper is organized as follows. In Section 2, we discuss related work and provide some context for this paper. In Section 3 we introduce the SOFT-IMPUTE algorithm and study its convergence properties in Section 4. The computational aspects of the algorithm are described in Section 5, and Section 6 discusses how nuclear norm regularization can be generalized to more aggressive and general types of spectral regularization. Section 7 describes post-processing of "selectors" and initialization. We discuss comparisons with related work, simulations and experimental studies in Section 9 and application to the Netflix data in Section 10.

2. Context and Related Work

Candès and Tao (2009), Cai et al. (2008), and Candès and Recht (2008) consider the criterion

$$\begin{aligned} &\text{minimize} && \|Z\|_* \\ &\text{subject to} && Z_{ij} = X_{ij},\ \forall (i,j) \in \Omega. \end{aligned} \tag{4}$$

With $\delta = 0$, the criterion (1) is equivalent to (4), in that it requires the training error to be zero. Cai et al. (2008) propose a first-order singular-value-thresholding algorithm SVT scalable to large matrices for the problem (4). They comment on the problem (2) with $\delta > 0$, but dismiss it as being computationally prohibitive for large problems.

We believe that (4) will almost always be too rigid and will result in over-fitting. If minimization of prediction error is an important goal, then the optimal solution $\hat{Z}$ will typically lie somewhere in the interior of the path indexed by δ (Figures 2, 3 and 4).

In this paper we provide an algorithm SOFT-IMPUTE for computing solutions of (3) on a grid of λ values, based on warm restarts. The algorithm is inspired by SVD-IMPUTE (Troyanskaya et al.,

1. For large problems data transfer, access and reading take quite a lot of time and is dependent upon the platform and machine. Over here we report the times taken for the computational bottle-neck, that is, the SVD computations over all iterations. All times are reported based on computations done in a Intel Xeon Linux 3GHz processor using MATLAB, with no C or Fortran interlacing.

2001)—an EM-type (Dempster et al., 1977) iterative algorithm that alternates between imputing the missing values from a current SVD, and updating the SVD using the "complete" data matrix. In its very motivation, SOFT-IMPUTE is different from generic first order algorithms (Cai et al., 2008; Ma et al.; Ji and Ye, 2009). The latter require the specification of a step size, and can be quite sensitive to the chosen value. Our algorithm does not require a step-size, or any such parameter.

The iterative algorithms proposed in Ma et al. and Ji and Ye (2009) require the computation of a SVD of a dense matrix (with dimensions equal to the size of the matrix X) at every iteration, as the bottleneck. This makes the algorithms prohibitive for large scale computations. Ma et al. use randomized algorithms for the SVD computation. Our algorithm SOFT-IMPUTE also requires an SVD computation at every iteration, but by exploiting the *problem structure*, can easily handle matrices of very large dimensions. At each iteration the non-sparse matrix has the structure:

$$Y = Y_{SP} \ \text{(Sparse)} \quad + \quad Y_{LR} \ \text{(Low Rank)}. \tag{5}$$

In (5) Y_{SP} has the same sparsity structure as the observed X, and Y_{LR} has rank $\tilde{r} \ll m,n$, where $\tilde{r}$ is very close to $r \ll m,n$ the rank of the estimated matrix Z (upon convergence of the algorithm). For large scale problems, we use iterative methods based on Lanczos bidiagonalization with partial re-orthogonalization (as in the PROPACK algorithm, Larsen, 1998), for computing the first $\tilde{r}$ singular vectors/values of Y. Due to the specific structure of (5), multiplication by Y and Y' can both be achieved in a cost-efficient way. In decomposition (5), the computationally burdensome work in computing a low-rank SVD is of an order that depends linearly on the matrix dimensions. More precisely, evaluating each singular vector requires computation of the order of $O((m+n)\tilde{r}) + O(|\Omega|)$ flops and evaluating r' of them requires $O((m+n)\tilde{r}r') + O(|\Omega|r')$ flops. Exploiting warm-starts, we observe that $\tilde{r} \approx r$—hence every SVD step of our algorithm computes r singular vectors, with complexity of the order $O((m+n)r^2) + O(|\Omega|r)$ flops. This computation is performed for the number of iterations SOFT-IMPUTE requires to run till convergence or a certain tolerance.

In this paper we show asymptotic convergence of SOFT-IMPUTE and further derive its non-asymptotic rate of convergence which scales as $O(1/k)$ (k denotes the iteration number). However, in our experimental studies on low-rank matrix completion, we have observed that our algorithm is faster (based on timing comparisons) than the accelerated version of Nesterov (Ji and Ye, 2009; Nesterov, 2007), having a provable (worst case) convergence rate of $O(\frac{1}{k^2})$. With warm-starts SOFT-IMPUTE computes the entire regularization path very efficiently along a dense series of values for λ.

Although the nuclear norm is motivated here as a convex relaxation to a rank constraint, we believe in many situations it will outperform the rank-restricted estimator (1). This is supported by our experimental studies. We draw the natural analogy with model selection in linear regression, and compare best-subset regression (ℓ_0 regularization) with the LASSO (ℓ_1 regularization, Tibshirani, 1996; Hastie et al., 2009). There too the ℓ_1 penalty can be viewed as a convex relaxation of the ℓ_0 penalty. But in many situations with moderate sparsity, the LASSO will outperform best subset in terms of prediction accuracy (Friedman, 2008; Hastie et al., 2009; Mazumder et al., 2009). By shrinking the parameters in the model (and hence reducing their variance), the lasso permits more parameters to be included. The nuclear norm is the ℓ_1 penalty in matrix completion, as compared to the ℓ_0 rank. By shrinking the singular values, we allow more dimensions to be included without incurring undue estimation variance.

Another class of techniques used in collaborative filtering problems are close in spirit to (2). These are known as *maximum margin matrix factorization* methods—in short MMMF—and use

a factor model for the matrix Z (Srebro et al., 2005b). Let $Z = UV'$ where $U_{m\times r'}$ and $V_{n\times r'}$, and consider the following problem

$$\underset{U,V}{\text{minimize}}\, \frac{1}{2}\sum_{(i,j)\in\Omega}(X_{ij}-(UV')_{ij})^2+\frac{\lambda}{2}(\|U\|_F^2+\|V\|_F^2). \tag{6}$$

It turns out that (6) is intimately related to (3), since (see Lemma 6)

$$||Z||_* = \min_{U,V:\, Z=UV'} \frac{1}{2}\left(\|U\|_F^2+\|V\|_F^2\right).$$

For example, if $r' = \min(m,n)$, the solution to (6) coincides with the solution to (3).[2] However, (6) is not convex in its arguments, while (3) is. We compare these two criteria in detail in Section 8, and the relative performance of their respective algorithms in Section 9.2.

3. SOFT-IMPUTE–an Algorithm for Nuclear Norm Regularization

We first introduce some notation that will be used for the rest of this article.

3.1 Notation

We adopt the notation of Cai et al. (2008). Define a matrix $P_\Omega(Y)$ (with dimension $m\times n$)

$$P_\Omega(Y)\,(i,j) = \begin{cases} Y_{ij} & \text{if } (i,j)\in\Omega \\ 0 & \text{if } (i,j)\notin\Omega, \end{cases} \tag{7}$$

which is a projection of the matrix $Y_{m\times n}$ onto the observed entries. In the same spirit, define the complementary projection $P_\Omega^\perp(Y)$ via $P_\Omega^\perp(Y)+P_\Omega(Y)=Y$. Using (7) we can rewrite $\sum_{(i,j)\in\Omega}(X_{ij}-Z_{ij})^2$ as $\|P_\Omega(X)-P_\Omega(Z)\|_F^2$.

3.2 Nuclear Norm Regularization

We present the following lemma, which forms a basic ingredient in our algorithm.

Lemma 1 *Suppose the matrix $W_{m\times n}$ has rank r. The solution to the optimization problem*

$$\underset{Z}{\text{minimize}} \quad \frac{1}{2}\|W-Z\|_F^2+\lambda\|Z\|_* \tag{8}$$

is given by $\hat{Z} = \mathbf{S}_\lambda(W)$ where

$$\mathbf{S}_\lambda(W) \equiv UD_\lambda V' \quad \textit{with} \quad D_\lambda = \text{diag}\left[(d_1-\lambda)_+,\ldots,(d_r-\lambda)_+\right], \tag{9}$$

UDV' is the SVD of W, $D=\text{diag}\,[d_1,\ldots,d_r]$, and $t_+ = \max(t,0)$.

The notation $\mathbf{S}_\lambda(W)$ refers to *soft-thresholding* (Donoho et al., 1995). Lemma 1 appears in Cai et al. (2008) and Ma et al. where the proof uses the sub-gradient characterization of the nuclear norm. In Appendix A.1 we present an entirely different proof, which can be extended in a relatively straightforward way to other complicated forms of spectral regularization discussed in Section 6. Our proof is followed by a remark that covers these more general cases.

2. We note here that the original MMMF formulation uses $r' = \min\{m,n\}$. In this paper we will consider it for a family of r' values.

3.3 Algorithm

Using the notation in 3.1, we rewrite (3) as:

$$\underset{Z}{\text{minimize}} \quad f_\lambda(Z) := \frac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_F^2 + \lambda\|Z\|_* \,. \tag{10}$$

We now present Algorithm 1—SOFT-IMPUTE—for computing a series of solutions to (10) for different values of λ using warm starts.

Algorithm 1 SOFT-IMPUTE

1. Initialize $Z^{\text{old}} = 0$.
2. Do for $\lambda_1 > \lambda_2 > \ldots > \lambda_K$:
   (a) Repeat:
      i. Compute $Z^{\text{new}} \leftarrow \mathbf{S}_{\lambda_k}(P_\Omega(X) + P_\Omega^\perp(Z^{\text{old}}))$.
      ii. If $\frac{\|Z^{\text{new}} - Z^{\text{old}}\|_F^2}{\|Z^{\text{old}}\|_F^2} < \varepsilon$ exit.
      iii. Assign $Z^{\text{old}} \leftarrow Z^{\text{new}}$.
   (b) Assign $\hat{Z}_{\lambda_k} \leftarrow Z^{\text{new}}$.
3. Output the sequence of solutions $\hat{Z}_{\lambda_1}, \ldots, \hat{Z}_{\lambda_K}$.

The algorithm repeatedly replaces the missing entries with the current guess, and then updates the guess by solving (8). Figures 2, 3 and 4 show some examples of solutions using SOFT-IMPUTE (blue continuous curves). We see test and training error in the top rows as a function of the nuclear norm, obtained from a grid of values Λ. These error curves show a smooth and very competitive performance.

4. Convergence Analysis

In this section we study the convergence properties of Algorithm 1. Unlike generic first-order methods (Nesterov, 2003) including competitive first-order methods for nuclear norm regularized problems (Cai et al., 2008; Ma et al.), SOFT-IMPUTE does not involve the choice of any additional step-size. Most importantly our algorithm is readily scalable for solving large scale semidefinite programming problems (2) and (10) as will be explained later in Section 5.

For an arbitrary matrix $\tilde{Z}$, define

$$Q_\lambda(Z|\tilde{Z}) = \frac{1}{2}\|P_\Omega(X) + P_\Omega^\perp(\tilde{Z}) - Z\|_F^2 + \lambda\|Z\|_* \tag{11}$$

as a surrogate of the objective function $f_\lambda(z)$. Note that $f_\lambda(\tilde{Z}) = Q_\lambda(\tilde{Z}|\tilde{Z})$ for any $\tilde{Z}$.

In Section 4.1, we show that the sequence Z_λ^k generated via SOFT-IMPUTE *converges* asymptotically, that is, as $k \to \infty$ to a minimizer of the objective function $f_\lambda(Z)$. SOFT-IMPUTE produces a sequence of solutions for which the criterion decreases to the optimal solution with every iteration and the successive iterates get closer to the optimal set of solutions of the problem 10. Section 4.2

derives the non-asymptotic convergence rate of the algorithm. The latter analysis concentrates on the objective values $f_\lambda(Z_\lambda^k)$. Due to computational resources if one wishes to stop the algorithm after K iterations, then Theorem 2 provides a certificate of how *far* Z_λ^k is from the solution. Though Section 4.1 alone establishes the convergence of $f_\lambda(Z_\lambda^k)$ to the minimum of $f_\lambda(Z)$, this does not, in general, settle the *convergence* of Z_λ^k unless further conditions (like strong convexity) are imposed on $f_\lambda(\cdot)$.

4.1 Asymptotic Convergence

Lemma 2 *For every fixed $\lambda \geq 0$, define a sequence Z_λ^k by*

$$Z_\lambda^{k+1} = \arg\min_Z Q_\lambda(Z|Z_\lambda^k)$$

with any starting point Z_λ^0. The sequence Z_λ^k satisfies

$$f_\lambda(Z_\lambda^{k+1}) \leq Q_\lambda(Z_\lambda^{k+1}|Z_\lambda^k) \leq f_\lambda(Z_\lambda^k).$$

Proof Note that

$$Z_\lambda^{k+1} = \mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k)). \tag{12}$$

By Lemma 1 and the definition (11) of $Q_\lambda(Z|Z_\lambda^k)$, we have:

$$\begin{aligned}
f_\lambda(Z_\lambda^k) &= Q_\lambda(Z_\lambda^k|Z_\lambda^k) \\
&= \frac{1}{2}\|P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k) - Z_\lambda^k\|_F^2 + \lambda\|Z_\lambda^k\|_* \\
&\geq \min_Z \frac{1}{2}\left\{\|P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k) - Z\|_F^2\right\} + \lambda\|Z\|_* \\
&= Q_\lambda(Z_\lambda^{k+1}|Z_\lambda^k) \\
&= \frac{1}{2}\|\left\{P_\Omega(X) - P_\Omega(Z_\lambda^{k+1})\right\} + \left\{P_\Omega^\perp(Z_\lambda^k) - P_\Omega^\perp(Z_\lambda^{k+1})\right\}\|_F^2 + \lambda\|Z_\lambda^{k+1}\|_* \\
&= \frac{1}{2}\left\{\|P_\Omega(X) - P_\Omega(Z_\lambda^{k+1})\|_F^2 + \|P_\Omega^\perp(Z_\lambda^k) - P_\Omega^\perp(Z_\lambda^{k+1})\|_F^2\right\} + \lambda\|Z_\lambda^{k+1}\|_* \qquad (13) \\
&\geq \frac{1}{2}\|P_\Omega(X) - P_\Omega(Z_\lambda^{k+1})\|_F^2 + \lambda\|Z_\lambda^{k+1}\|_* \qquad (14) \\
&= Q_\lambda(Z_\lambda^{k+1}|Z_\lambda^{k+1}) \\
&= f(Z_\lambda^{k+1}).
\end{aligned}$$

■

Lemma 3 *The nuclear norm shrinkage operator $\mathbf{S}_\lambda(\cdot)$ satisfies the following for any W_1, W_2 (with matching dimensions)*

$$\|\mathbf{S}_\lambda(W_1) - \mathbf{S}_\lambda(W_2)\|_F^2 \leq \|W_1 - W_2\|_F^2.$$

In particular this implies that $\mathbf{S}_\lambda(W)$ is a continuous map in W.

Lemma 3 is proved in Ma et al.; their proof is complex and based on trace inequalities. We give a concise proof based on elementary convex analysis in Appendix A.2.

Lemma 4 *The successive differences $\|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2$ of the sequence Z_λ^k are monotone decreasing:*

$$\|Z_\lambda^{k+1} - Z_\lambda^k\|_F^2 \leq \|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2 \quad \forall k. \tag{15}$$

Moreover the difference sequence converges to zero. That is

$$Z_\lambda^{k+1} - Z_\lambda^k \to 0 \text{ as } k \to \infty.$$

The proof of Lemma 4 is given in Appendix A.3.

Lemma 5 *Every limit point of the sequence Z_λ^k defined in Lemma 2 is a stationary point of*

$$\frac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_F^2 + \lambda\|Z\|_*.$$

Hence it is a solution to the fixed point equation

$$Z = \mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z)). \tag{16}$$

The proof of Lemma 5 is given in Appendix A.4.

Theorem 1 *The sequence Z_λ^k defined in Lemma 2 converges to a limit Z_λ^∞ that solves*

$$\underset{Z}{\text{minimize}} \quad \frac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_F^2 + \lambda\|Z\|_*. \tag{17}$$

Proof It suffices to prove that Z_λ^k converges; the theorem then follows from Lemma 5.

Let $\hat{Z}_\lambda$ be a limit point of the sequence Z_λ^k. There exists a subsequence m_k such that $Z_\lambda^{m_k} \to \hat{Z}_\lambda$. By Lemma 5, $\hat{Z}_\lambda$ solves the problem (17) and satisfies the fixed point equation (16).

Hence

$$\begin{aligned}
\|\hat{Z}_\lambda - Z_\lambda^k\|_F^2 &= \|\mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(\hat{Z}_\lambda)) - \mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}))\|_F^2 && (18)\\
&\leq \|(P_\Omega(X) + P_\Omega^\perp(\hat{Z}_\lambda)) - (P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}))\|_F^2\\
&= \|P_\Omega^\perp(\hat{Z}_\lambda - Z_\lambda^{k-1})\|_F^2\\
&\leq \|\hat{Z}_\lambda - Z_\lambda^{k-1}\|_F^2. && (19)
\end{aligned}$$

In (18) two substitutions were made; the left one using (16) in Lemma 5, the right one using (12). Inequality (19) implies that the sequence $\|\hat{Z}_\lambda - Z_\lambda^{k-1}\|_F^2$ converges as $k \to \infty$. To show the convergence of the sequence Z_λ^k it suffices to prove that the sequence $\hat{Z}_\lambda - Z_\lambda^k$ converges to zero. We prove this by contradiction.

Suppose the sequence Z_λ^k has another limit point $Z_\lambda^+ \neq \hat{Z}_\lambda$. Then $\hat{Z}_\lambda - Z_\lambda^k$ has two distinct limit points 0 and $Z_\lambda^+ - \hat{Z}_\lambda \neq 0$. This contradicts the convergence of the sequence $\|\hat{Z}_\lambda - Z_\lambda^{k-1}\|_F^2$. Hence the sequence Z_λ^k converges to $\hat{Z}_\lambda := Z_\lambda^\infty$. ■

The inequality in (19) implies that at every iteration Z_λ^k gets closer to an optimal solution for the problem (17).[3] This property holds in addition to the decrease of the objective function (Lemma 2) at every iteration.

3. In fact this statement can be strengthened further—at every iteration the distance of the estimate decreases from the set of optimal solutions.

4.2 Convergence Rate

In this section we derive the worst case convergence rate of SOFT-IMPUTE.

Theorem 2 *For every fixed $\lambda \geq 0$, the sequence $Z_\lambda^k; k \geq 0$ defined in Lemma 2 has the following non-asymptotic (worst) rate of convergence:*

$$f_\lambda(Z_\lambda^k) - f_\lambda(Z_\lambda^\infty) \leq \frac{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2}{k+1}. \tag{20}$$

The proof of this theorem is in Appendix A.6.

In light of Theorem 2, a $\delta > 0$ accurate solution of $f_\lambda(Z)$ is obtained after a maximum of $\frac{2}{\delta}\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2$ iterations. Using warm-starts, SOFT-IMPUTE traces out the path of solutions on a grid of λ values $\lambda_1 > \lambda_2 > \ldots > \lambda_K$ with a total of [4]

$$\sum_{i=1}^{K} \frac{2}{\delta}\|\hat{Z}_{\lambda_{i-1}} - Z_{\lambda_i}^\infty\|_F^2 \tag{21}$$

iterations. Here $\hat{Z}_{\lambda_0} = 0$ and $\hat{Z}_{\lambda_i}$ denotes the output of SOFT-IMPUTE (upon convergence) for $\lambda = \lambda_i$ ($i \in \{1, \ldots, K-1\}$). The solutions $Z_{\lambda_i}^\infty$ and $Z_{\lambda_{i-1}}^\infty$ are likely to be *close* to each other, especially on a dense grid of λ_i's. Hence every summand of (21) and the total number of iterations is expected to be significantly smaller than that obtained via arbitrary cold-starts.

5. Computational Complexity

The computationally demanding part of Algorithm 1 is in $\mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k))$. This requires calculating a low-rank SVD of a matrix, since the underlying model assumption is that $\text{rank}(Z) \ll \min\{m,n\}$. In Algorithm 1, for fixed λ, the entire sequence of matrices Z_λ^k have explicit[5] low-rank representations of the form $U_k D_k V_k'$ corresponding to $\mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}))$.

In addition, observe that $P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k)$ can be rewritten as

$$\begin{array}{rcccc} P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k) & = & \{P_\Omega(X) - P_\Omega(Z_\lambda^k)\} & + & Z_\lambda^k \\ & = & \textit{Sparse} & + & \textit{Low Rank}. \end{array} \tag{22}$$

In the numerical linear algebra literature, there are very efficient direct matrix factorization methods for calculating the SVD of matrices of moderate size (at most a few thousand). When the matrix is sparse, larger problems can be solved but the computational cost depends heavily upon the sparsity structure of the matrix. In general however, for large matrices one has to resort to indirect iterative methods for calculating the leading singular vectors/values of a matrix. There is a lot research in numerical linear algebra for developing sophisticated algorithms for this purpose. In this paper we will use the PROPACK algorithm (Larsen, 2004, 1998) because of its low storage requirements, effective flop count and its well documented MATLAB version. The algorithm for calculating the truncated SVD for a matrix W (say), becomes efficient if multiplication operations Wb_1 and $W'b_2$ (with $b_1 \in \Re^n$, $b_2 \in \Re^m$) can be done with minimal cost.

4. We assume the solution $\hat{Z}_\lambda$ at every $\lambda \in \{\lambda_1, \ldots, \lambda_K\}$ is computed to an accuracy of $\delta > 0$.

5. Though we cannot prove theoretically that every iterate of the sequence Z_λ^k will be of low-rank; this observation is rather practical based on the manner in which we trace out the entire path of solutions based on warm-starts. Our simulation results support this observation as well.

Algorithm SOFT-IMPUTE requires repeated computation of a truncated SVD for a matrix W with structure as in (22). Assume that at the current iterate, the matrix Z_λ^k has rank $\tilde{r}$. Note that in (22) the term $P_\Omega(Z_\lambda^k)$ can be computed in $O(|\Omega|\tilde{r})$ flops using only the required outer products (i.e., our algorithm does not compute the matrix explicitly).

The cost of computing the truncated SVD will depend upon the cost in the operations Wb_1 and $W'b_2$ (which are equal). For the sparse part these multiplications cost $O(|\Omega|)$. Although it costs $O(|\Omega|\tilde{r})$ to create the matrix $P_\Omega(Z_\lambda^k)$, this is used for each of the $\tilde{r}$ such multiplications (which also cost $O(|\Omega|\tilde{r})$), so we need not include that cost here. The Low Rank part costs $O((m+n)\tilde{r})$ for the multiplication by b_1. Hence the cost is $O(|\Omega|)+O((m+n)\tilde{r})$ per vector multiplication. Supposing we want a $\tilde{r}$ rank SVD of the matrix (22), the cost will be of the order of $O(|\Omega|\tilde{r})+O((m+n)(\tilde{r})^2)$ (for that iteration, that is, to obtain Z_λ^{k+1} from Z_λ^k). Suppose the rank of the solution Z_λ^k is r, then in light of our above observations $\tilde{r} \approx r \ll \min\{m,n\}$ and the order is $O(|\Omega|r)+O((m+n)r^2)$.

For the reconstruction problem to be theoretically meaningful in the sense of Candès and Tao (2009) we require that $|\Omega| \approx nr \cdot \text{poly}(\log n)$. In practice often $|\Omega|$ is very small. Hence introducing the *Low Rank* part does not add any further complexity in the multiplication by W and W'. So the dominant cost in calculating the truncated SVD in our algorithm is $O(|\Omega|)$. The SVT algorithm (Cai et al., 2008) for exact matrix completion (4) involves calculating the SVD of a sparse matrix with cost $O(|\Omega|)$. This implies that the computational order of SOFT-IMPUTE and that of SVT is the same. This order computation does not include the number of iterations required for convergence. In our experimental studies we use warm-starts for efficiently computing the entire regularization path. On small scale examples, based on comparisons with the accelerated gradient method of Nesterov (see Section 9.3; Ji and Ye, 2009; Nesterov, 2007) we find that our algorithm converges faster than the latter in terms of run-time and number of SVD computations/ iterations. This supports the computational effectiveness of SOFT-IMPUTE. In addition, since the true rank of the matrix $r \ll \min\{m,n\}$, the computational cost of evaluating the truncated SVD (with rank $\approx r$) is linear in matrix dimensions. This justifies the large-scale computational feasibility of our algorithm.

The above discussions focus on the computational complexity for obtaining a low-rank SVD, which is to be performed at every iteration of SOFT-IMPUTE. Similar to the total iteration complexity bound of SOFT-IMPUTE (21), the total cost to compute the regularization path on a grid of λ values is given by:

$$\sum_{i=1}^{K} O\left((|\Omega|\bar{r}_{\lambda_i} + (m+n)\bar{r}_{\lambda_i}^2)\frac{2}{\delta}\|\hat{Z}_{\lambda_{i-1}} - Z_{\lambda_i}^{\infty}\|_F^2 \right).$$

Here $\bar{r}_\lambda$ denotes the rank[6] (on an average) of the iterates Z_λ^k generated by SOFT-IMPUTE for fixed λ.

The PROPACK package does not allow one to request (and hence compute) only the singular values larger than a threshold λ—one has to specify the number in advance. So once all the computed singular values fall above the current threshold λ, our algorithm increases the number to be computed until the smallest is smaller than λ. In large scale problems, we put an absolute limit on the maximum number.

6. We assume, above that the grid of values $\lambda_1 > \ldots \lambda_K$ is such that *all* the solutions $Z_\lambda, \lambda \in \{\lambda_1, \ldots, \lambda_K\}$ are of *small* rank, as they appear in Section 5.

6. Generalized Spectral Regularization: From Soft to Hard Thresholding

In Section 1 we discussed the role of the nuclear norm as a convex surrogate for the rank of a matrix, and drew the analogy with LASSO regression versus best-subset selection. We argued that in many problems ℓ_1 regularization gives better prediction accuracy. However, if the underlying model is very sparse, then the LASSO with its uniform shrinkage can both overestimate the number of non-zero coefficients (Friedman, 2008) in the model, and overly shrink (bias) those included toward zero. In this section we propose a natural generalization of SOFT-IMPUTE to overcome these problems.

Consider again the problem

$$\underset{\text{rank}(Z)\le k}{\text{minimize}} \ \frac{1}{2}\|P_\Omega(X)-P_\Omega(Z)\|_F^2,$$

a rephrasing of (1). This best rank-k solution also solves

$$\text{minimize}\,\frac{1}{2}\|P_\Omega(X)-P_\Omega(Z)\|_F^2+\lambda\sum_j I(\gamma_j(Z)>0),$$

where $\gamma_j(Z)$ is the jth singular value of Z, and for a suitable choice of λ that produces a solution with rank k.

The "fully observed" matrix version of the above problem is given by the ℓ_0 version of (8) as follows:

$$\underset{Z}{\text{minimize}} \ \frac{1}{2}\|W-Z\|_F^2+\lambda\|Z\|_0, \tag{23}$$

where $\|Z\|_0=\text{rank}(Z)$. The solution of (23) is given by a reduced-rank SVD of W; for every λ there is a corresponding $q=q(\lambda)$ number of singular-values to be retained in the SVD decomposition. Problem (23) is non-convex in W but its global minimizer can be evaluated. As in (9) the thresholding operator resulting from (23) is

$$\mathbf{S}_\lambda^H(W)=UD_qV' \quad \text{where} \quad D_q=\text{diag}\,(d_1,\ldots,d_q,0,\ldots,0)\,.$$

Similar to SOFT-IMPUTE (Algorithm 1), we present below HARD-IMPUTE (Algorithm 2) for the ℓ_0 penalty. The continuous parameterization via λ does not appear to offer obvious advantages over rank-truncation methods. We note that it does allow for a continuum of warm starts, and is a natural post-processor for the output of SOFT-IMPUTE (next section). But it also allows for further generalizations that bridge the gap between hard and soft regularization methods.

In penalized regression there have been recent developments directed towards "bridging" the gap between the ℓ_1 and ℓ_0 penalties (Friedman, 2008; Zhang, 2010; Mazumder et al., 2009). This is done via using non-convex penalties that are a better surrogate (in the sense of approximating the penalty) to ℓ_0 over the ℓ_1. They also produce less biased estimates than those produced by the ℓ_1 penalized solutions. When the underlying model is very sparse they often perform very well, and enjoy superior prediction accuracy when compared to softer penalties like ℓ_1. These methods still shrink, but are less aggressive than the best-subset selection.

By analogy, we propose using a more sophisticated version of spectral regularization. This goes beyond nuclear norm regularization by using slightly more aggressive penalties that bridge the gap between ℓ_1 (nuclear norm) and ℓ_0(rank constraint). We propose minimizing

$$f_{\mathbf{p},\lambda}(Z) \ = \ \frac{1}{2}\|P_\Omega(X)-P_\Omega(Z)\|_F^2+\lambda\sum_j \mathbf{p}(\gamma_j(Z);\mu), \tag{24}$$

Algorithm 2 HARD-IMPUTE

1. Initialize $\tilde{Z}_{\lambda_k}$ $k = 1, \ldots, K$ (for example, using SOFT-IMPUTE; see Section 7).
2. Do for $\lambda_1 > \lambda_2 > \ldots > \lambda_K$:
   (a) Repeat:
      i. Compute $Z^{\text{new}} \leftarrow \mathbf{S}^H_{\lambda_k}(P_\Omega(X) + P^\perp_\Omega(Z^{\text{old}}))$.
      ii. If $\frac{\|Z^{\text{new}} - Z^{\text{old}}\|_F^2}{\|Z^{\text{old}}\|_F^2} < \varepsilon$ exit.
      iii. Assign $Z^{\text{old}} \leftarrow Z^{\text{new}}$.
   (b) Assign $\hat{Z}_{H,\lambda_k} \leftarrow Z^{\text{new}}$.
3. Output the sequence of solutions $\hat{Z}_{H,\lambda_1}, \ldots, \hat{Z}_{H,\lambda_K}$.

where $\mathbf{p}(|t|;\mu)$ is concave in $|t|$. The parameter $\mu \in [\mu_{\text{inf}}, \mu_{\text{sup}}]$ controls the degree of concavity. We may think of $p(|t|;\mu_{\text{inf}}) = |t|$ (ℓ_1 penalty) on one end and $p(|t|;\mu_{\text{sup}}) = \|t\|_0$ (ℓ_0 penalty) on the other. In particular for the ℓ_0 penalty denote $f_{\mathbf{p},\lambda}(Z)$ by $f_{H,\lambda}(Z)$ for "hard" thresholding. See Friedman (2008), Mazumder et al. (2009) and Zhang (2010) for examples of such penalties.

In Remark 1 in Appendix A.1 we argue how the proof can be modified for general types of spectral regularization. Hence for minimizing the objective (24) we will look at the analogous version of (8, 23) which is

$$\underset{Z}{\text{minimize}} \quad \frac{1}{2}\|W - Z\|_F^2 + \lambda \sum_j \mathbf{p}(\gamma_j(Z);\mu).$$

The solution is given by a thresholded SVD of W,

$$\mathbf{S}^{\mathbf{p}}_\lambda(W) = U D_{\mathbf{p},\lambda} V',$$

where $D_{\mathbf{p},\lambda}$ is a entry-wise thresholding of the diagonal entries of the matrix D consisting of singular values of the matrix W. The exact form of the thresholding depends upon the form of the penalty function $\mathbf{p}(\cdot;\cdot)$, as discussed in Remark 1. Algorithm 1 and Algorithm 2 can be modified for the penalty $\mathbf{p}(\cdot;\mu)$ by using a more general thresholding function $\mathbf{S}^{\text{p}}_\lambda(\cdot)$ in Step 2(a)i. The corresponding step becomes:

$$Z^{\text{new}} \leftarrow \mathbf{S}^{\mathbf{p}}_\lambda(P_\Omega(X) + P^\perp_\Omega(Z^{\text{old}})).$$

However these types of spectral regularization make the criterion (24) non-convex and hence it becomes difficult to optimize globally. Recht et al. (2007) and Bach (2008) also consider the rank estimation problem from a theoretical standpoint.

7. Post-processing of "Selectors" and Initialization

Because the ℓ_1 norm regularizes by shrinking the singular values, the number of singular values retained (through cross-validation, say) may exceed the actual rank of the matrix. In such cases it is reasonable to *undo* the shrinkage of the chosen models, which might permit a lower-rank solution.

If Z_λ is the solution to (10), then its *post-processed* version Z_λ^u obtained by "unshrinking" the eigen-values of the matrix Z_λ is obtained by

$$\begin{aligned} \alpha &= \underset{\alpha_i \geq 0,\, i=1,\ldots,r_\lambda}{\arg\min} \quad \|P_\Omega(X) - \sum_{i=1}^{r_\lambda} \alpha_i P_\Omega(u_i v_i')\|^2 \\ Z_\lambda^u &= UD_\alpha V', \end{aligned} \tag{25}$$

where $D_\alpha = \mathrm{diag}(\alpha_1, \ldots, \alpha_{r_\lambda})$. Here r_λ is the rank of Z_λ and $Z_\lambda = UD_\lambda V'$ is its SVD. The estimation in (25) can be done via ordinary least squares, which is feasible because of the sparsity of $P_\Omega(u_i v_i')$ and that r_λ is small.[7] If the least squares solutions α do not meet the positivity constraints, then the negative sign can be absorbed into the corresponding singular vector.

Rather than estimating a diagonal matrix D_α as above, one can insert a matrix $M_{r_\lambda \times r_\lambda}$ between U and V above to obtain better training error for the same rank. Hence given U, V (each of rank r_λ) from the SOFT-IMPUTE algorithm, we solve

$$\begin{aligned} \hat{M} &= \underset{M}{\arg\min} \quad \|P_\Omega(X) - P_\Omega(UMV')\|^2, \\ \text{where, } \hat{Z}_\lambda &= U\hat{M}V'. \end{aligned} \tag{26}$$

The objective function in (26) is the Frobenius norm of an affine function of M and hence can be optimized very efficiently. Scalability issues pertaining to the optimization problem (26) can be handled fairly efficiently via conjugate gradients. Criterion (26) will definitely lead to a decrease in training error as that attained by $\hat{Z} = UD_\lambda V'$ for the same rank and is potentially an attractive proposal for the original problem (1). However this heuristic cannot be caste as a (jointly) convex problem in (U, M, V). In addition, this requires the estimation of up to r_λ^2 parameters, and has the potential for over-fitting. In this paper we report experiments based on (25).

In many simulated examples we have observed that this post-processing step gives a good estimate of the underlying true rank of the matrix (based on prediction error). Since fixed points of Algorithm 2 correspond to local minima of the function (24), well-chosen warm starts $\tilde{Z}_\lambda$ are helpful. A reasonable prescription for warms-starts is the nuclear norm solution via (SOFT-IMPUTE), or the post processed version (25). The latter appears to significantly speed up convergence for HARD-IMPUTE. This observation is based on our simulation studies.

8. Soft-Impute and Maximum-Margin Matrix Factorization

In this section we compare in detail the MMMF criterion (6) with the SOFT-IMPUTE criterion (3). For ease of comparison here, we put down these criteria again using our P_Ω notation.

MMMF solves

$$\underset{U,V}{\text{minimize}} \quad \frac{1}{2}||P_\Omega(X - UV)||_F^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2), \tag{27}$$

where $U_{m\times r'}$ and $V_{n\times r'}$ are arbitrary (non-orthogonal) matrices. This problem formulation and related optimization methods have been explored by Srebro et al. (2005b) and Rennie and Srebro (2005).

7. Observe that the $P_\Omega(u_i v_i')$, $i = 1, \ldots, r_\lambda$ are not orthogonal, though the $u_i v_i'$ are.

SOFT-IMPUTE solves

$$\underset{Z}{\text{minimize}} \quad \frac{1}{2}||P_\Omega(X-Z)||_F^2 + \lambda\|Z\|_*. \tag{28}$$

For each given maximum rank, MMMF produces an estimate by doing further shrinkage with its quadratic regularization. SOFT-IMPUTE performs rank reduction and shrinkage at the same time, in one smooth convex operation. The following theorem shows that this one-dimensional SOFT-IMPUTE family lies exactly in the two-dimensional MMMF family.

Theorem 3 *Let X be $m \times n$ with observed entries indexed by Ω.*

1. *Let $r' = \min(m,n)$. Then the solutions to (27) and (28) coincide for all $\lambda \geq 0$.*
2. *Suppose $\hat{Z}^*$ is a solution to (28) for $\lambda^* > 0$, and let r^* be its rank. Then for any solution $\hat{U}, \hat{V}$ to (27) with $r' = r^*$ and $\lambda = \lambda^*$, $\hat{U}\hat{V}^T$ is a solution to (28). The SVD factorization of $\hat{Z}^*$ provides one such solution to (27). This implies that the solution space of (28) is contained in that of (27).*

Remarks:

1. Part 1 of this theorem appears in a slightly different form in Srebro et al. (2005b).
2. In part 1, we could use $r' > \min(m,n)$ and get the same equivalence. While this might seem unnecessary, there may be computational advantages; searching over a bigger space might protect against local minima. Likewise in part 2, we could use $r' > r^*$ and achieve the same equivalence. In either case, no matter what r' we use, the solution matrices $\hat{U}$ and $\hat{V}$ have the same rank as $\hat{Z}$.
3. Let $\hat{Z}(\lambda)$ be a solution to (28) at λ. We conjecture that rank $[\hat{Z}(\lambda)]$ is monotone non-increasing in λ. If this is the case, then Theorem 3, part 2 can be further strengthened to say that for all $\lambda \geq \lambda^*$ and $r' = r^*$ the solutions of (27) coincide with that of (28).

The MMMF criterion (27) defines a two-dimensional family of models indexed by (r', λ), while the SOFT-IMPUTE criterion (28) defines a one-dimensional family. In light of Theorem 3, this family is a special path in the two-dimensional grid of solutions $[\hat{U}_{(r',\lambda)}, \hat{V}_{(r',\lambda)}]$. Figure 1 depicts the situation. Any MMMF model at parameter combinations above the red squares are redundant, since their fit is the same at the red square. However, in practice the red squares are not known to MMMF, nor is the actual rank of the solution. Further orthogonalization of $\hat{U}$ and $\hat{V}$ would be required to reveal the rank, which would only be approximate (depending on the convergence criterion of the MMMF algorithm).

Despite the equivalence of (27) and (28) when $r' = \min(m,n)$, the criteria are quite different. While (28) is a convex optimization problem in Z, (27) is a non-convex problem in the variables U, V and has possibly several local minima; see also Abernethy et al. (2009). It has been observed empirically and theoretically (Burer and Monteiro, 2005; Rennie and Srebro, 2005) that bi-convex methods used in the optimization of (27) can get stuck in sub-optimal local minima for a small value of r' or a poorly chosen starting point. For a large number of factors r' and large dimensions m,n the computational cost may be quite high (See also experimental studies in Section 9.2).

Criterion (28) is convex in Z for every value of λ, and it outputs the solution $\hat{Z}$ in the form of its soft-thresholded SVD, implying that the "factors" U, V are already orthogonal and the rank is known.

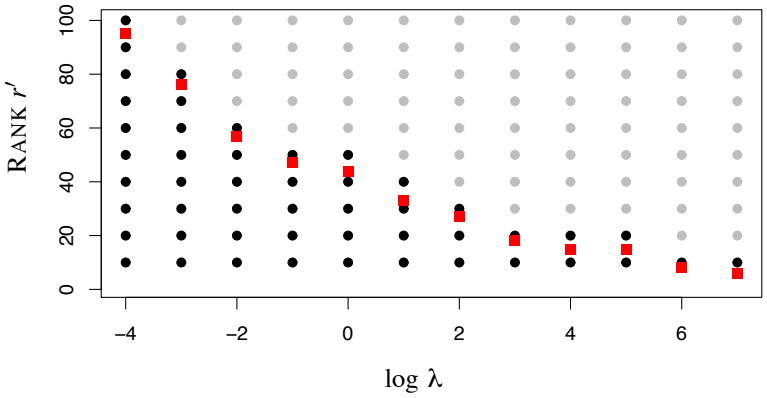


Figure 1: Comparison of the parameter space for MMMF (grey and black points), and SOFT-IMPUTE (red squares) for a simple example. Since all MMMF solutions with parameters above the red squares are identical to the SOFT-IMPUTE solutions at the red squares, all the grey points are redundant.

MMMF has two different tuning parameters r' and λ, both of which are related to the rank or spectral properties of the matrices U, V. SOFT-IMPUTE has only one tuning parameter λ. The presence of two tuning parameters is problematic:

- It results in a significant increase in computational burden, since for every given value of r', one needs to compute an entire system of solutions by varying λ (see Section 9 for illustrations).
- In practice when neither the optimal values of r' and λ are known, a two-dimensional search (for example, by cross validation) is required to select suitable values.

Further discussions and connections between the tuning parameters and spectral properties of the matrices can be found in Burer and Monteiro (2005) and Abernethy et al. (2009).

The proof of Theorem 3 requires a lemma.

Lemma 6 *For any matrix Z, the following holds:*

$$||Z||_* = \min_{U,V:\, Z=UV^T} \frac{1}{2}\left(\|U\|_F^2 + \|V\|_F^2\right). \tag{29}$$

If $\text{rank}(Z) = k \leq \min\{m,n\}$*, then the minimum above is attained at a factor decomposition* $Z = U_{m\times k}V_{n\times k}^T$.

Note that in the decomposition $Z = UV^T$ in (29) there is no constraint on the number of columns r of the factor matrices $U_{m\times r}$ and $V_{n\times r}$. Lemma 6 is stronger than similar results appearing in Rennie and Srebro (2005) and Abernethy et al. (2009) which establish (29) for $r = \min\{m,n\}$—we give a tighter estimate of the rank k of the underlying matrices. The proof is given in Appendix A.5.

8.1 Proof of Theorem 3

Part 1. For $r = \min(m,n)$, any matrix $Z_{m\times n}$ can be written in the form of $Z = UV^T$. The criterion (27) can be written as

$$\min_{U,V} \quad \tfrac{1}{2}||P_\Omega(X - UV^T)||_F^2 + \tfrac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \tag{30}$$

$$= \min_{U,V} \quad \tfrac{1}{2}||P_\Omega(X - UV^T)||_F^2 + \lambda\|UV^T\|_* \qquad \text{(by Lemma 6)}$$

$$= \min_{Z} \quad \tfrac{1}{2}||P_\Omega(X - Z)||_F^2 + \lambda\|Z\|_*. \tag{31}$$

The equivalence of the criteria in (30) and (31) completes the proof of part 1.
Part 2. Note that if we know that the solution $\hat{Z}^*$ to (28) with $\lambda = \lambda^*$ has rank r^*, then $\hat{Z}^*$ also solves

$$\min_{Z,\, \mathrm{rank}(Z)=r^*} \quad \tfrac{1}{2}||P_\Omega(X - Z)||_F^2 \quad + \lambda\|Z\|_*.$$

We now repeat the steps (30)—(31), restricting the rank r' of U and V to be $r' = r^*$, and the result follows. ■

9. Numerical Experiments and Comparisons

In this section we study the performance of SOFT-IMPUTE, its post-processed variants, and HARD-IMPUTE for noisy matrix completion problems. The examples assert our claim that the matrix reconstruction criterion (4) (Cai et al., 2008) is too rigid if one seeks good predictive models. We include the related procedures of Rennie and Srebro (2005) and Keshavan et al. (2009) in our comparisons.

The reconstruction algorithm OPTSPACE, described in Keshavan et al. (2009) considers criterion (1) (in the presence of noise). It uses the representation $Z = USV'$ (which need not correspond to the SVD). OPTSPACE alternates between estimating S and U,V (in a Grassmann manifold) for computing a rank-r decomposition $\hat{Z} = \hat{U}\hat{S}\hat{V}'$. It starts with a sparse SVD on a *clean* version of the observed matrix $P_\Omega(X)$. This is similar to the formulation of MMMF (27) as detailed in Section 8, without the squared Frobenius norm regularization on the components U,V.

To summarize, we study the following methods:

1. SOFT-IMPUTE–Algorithm 1;
2. SOFT-IMPUTE+–post-processing on the output of SOFT-IMPUTE, as in Section 7;
3. HARD-IMPUTE–Algorithm 2, starting with the output of SOFT-IMPUTE+;
4. SVT–algorithm by Cai et al. (2008);
5. OPTSPACE–reconstruction algorithm by Keshavan et al. (2009);
6. MMMF–algorithm for (6) as in Rennie and Srebro (2005).

In all our simulation studies we use the underlying model $Z_{m\times n} = U_{m\times r}V'_{r\times n} + \varepsilon$, where U and V are random matrices with standard normal Gaussian entries, and ε is i.i.d. Gaussian. Ω is uniformly random over the indices of the matrix with $p\%$ percent of missing entries. These are the models under which the coherence conditions hold true for the matrix completion problem to be meaningful (Candès and Tao, 2009; Keshavan et al., 2009). The signal to noise ratio for the model and the test-error (standardized) are defined as

$$\text{SNR} = \sqrt{\frac{\text{var}(UV')}{\text{var}(\varepsilon)}}; \quad \text{Test Error} = \frac{\|P_\Omega^\perp(UV' - \hat{Z})\|_F^2}{\|P_\Omega^\perp(UV')\|_F^2}.$$

Training error (standardized) is defined as

$$\text{Training Error} = \frac{\|P_\Omega(Z - \hat{Z})\|_F^2}{\|P_\Omega(Z)\|_F^2},$$

the fraction of the error explained on the observed entries by the estimate relative to a zero estimate.

Figures 2, 3 and 4 show training and test error for all of the algorithms mentioned above—both as a function of nuclear norm and rank—for the three problem instances. The results displayed in the figures are averaged over 50 simulations, and also show one-standard-error bands (hardly visible). In all examples $(m,n) = (100,100)$. For MMMF we use $r' = \min(m,n) = 100$, the number of columns in U and V. The performance of MMMF is displayed only in the plots with the nuclear norm along the horizontal axis, since the algorithm does not deliver a precise rank. SNR, true rank and percentage of missing entries are indicated in the figures. There is a unique correspondence between λ and nuclear norm. The plots versus rank indicate how effective the nuclear norm is as a rank approximation—that is whether it recovers the true rank while minimizing prediction error.

For routines not our own we use the MATLAB code as supplied on webpages by the authors. For SVT second author of Cai et al. (2008), for OPTSPACE third author of Keshavan et al. (2009), and for MMMF first author of Rennie and Srebro (2005).

9.1 Observations

The captions of each of Figures 2–4 detail the results, which we summarize here. For the first two figures, the noise is quite high with SNR$= 1$, and 50% of the entries are missing. In Figure 2 the true rank is 10, while in Figure 3 it is 6. SOFT-IMPUTE, MMMF and SOFT-IMPUTE+ have the best prediction performance, while SOFT-IMPUTE+ is better at estimating the correct rank. The other procedures perform poorly here, although OPTSPACE improves somewhat in Figure 3. SVT has very poor prediction error, suggesting once again that exactly fitting the training data is far too rigid. SOFT-IMPUTE+ has the best performance in Figure 3 (smaller rank—more aggressive fitting), and HARD-IMPUTE starts recovering here. In both figures the training error for SOFT-IMPUTE (and hence MMMF) wins as a function of nuclear norm (as it must, by construction), but the more aggressive fitters SOFT-IMPUTE+ and HARD-IMPUTE have better training error as a function of rank.

Though the nuclear norm is often viewed as a surrogate for the rank of a matrix, we see in these examples that it can provide a superior mechanism for regularization. This is similar to the performance of LASSO in the context of regression. Although the LASSO penalty can be viewed as a convex surrogate for the ℓ_0 penalty in model selection, its ℓ_1 penalty provides a smoother and often better basis for regularization.

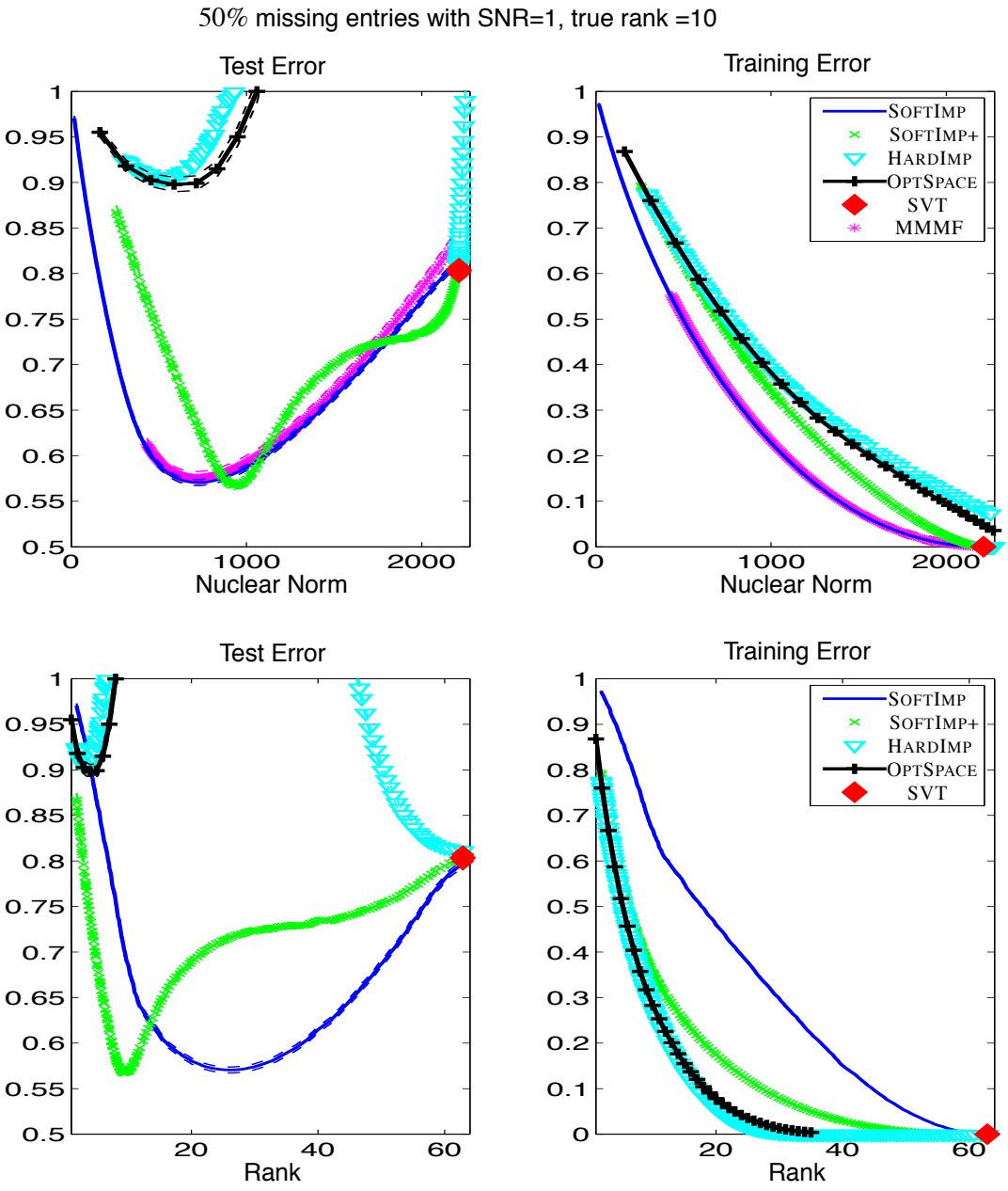


Figure 2: SOFTIMP+ refers to the post-processing after SOFT-IMPUTE; HARD-IMPUTE uses SOFT-IMP+ as starting values. Both SOFT-IMPUTE and SOFT-IMPUTE+ perform well (prediction error) in the presence of noise; the latter estimates the actual rank of the matrix. MMMF (with full rank 100 factor matrices) has performance similar to SOFT-IMPUTE. HARD-IMPUTE and OPTSPACE show poor prediction error. SVT also has poor prediction error, confirming our claim in this example that criterion (4) can result in overfitting; it recovers a matrix with high nuclear norm and rank > 60 where the true rank is only 10. Values of test error larger than one are not shown in the figure. OPTSPACE is evaluated for a series of ranks ≤ 30.

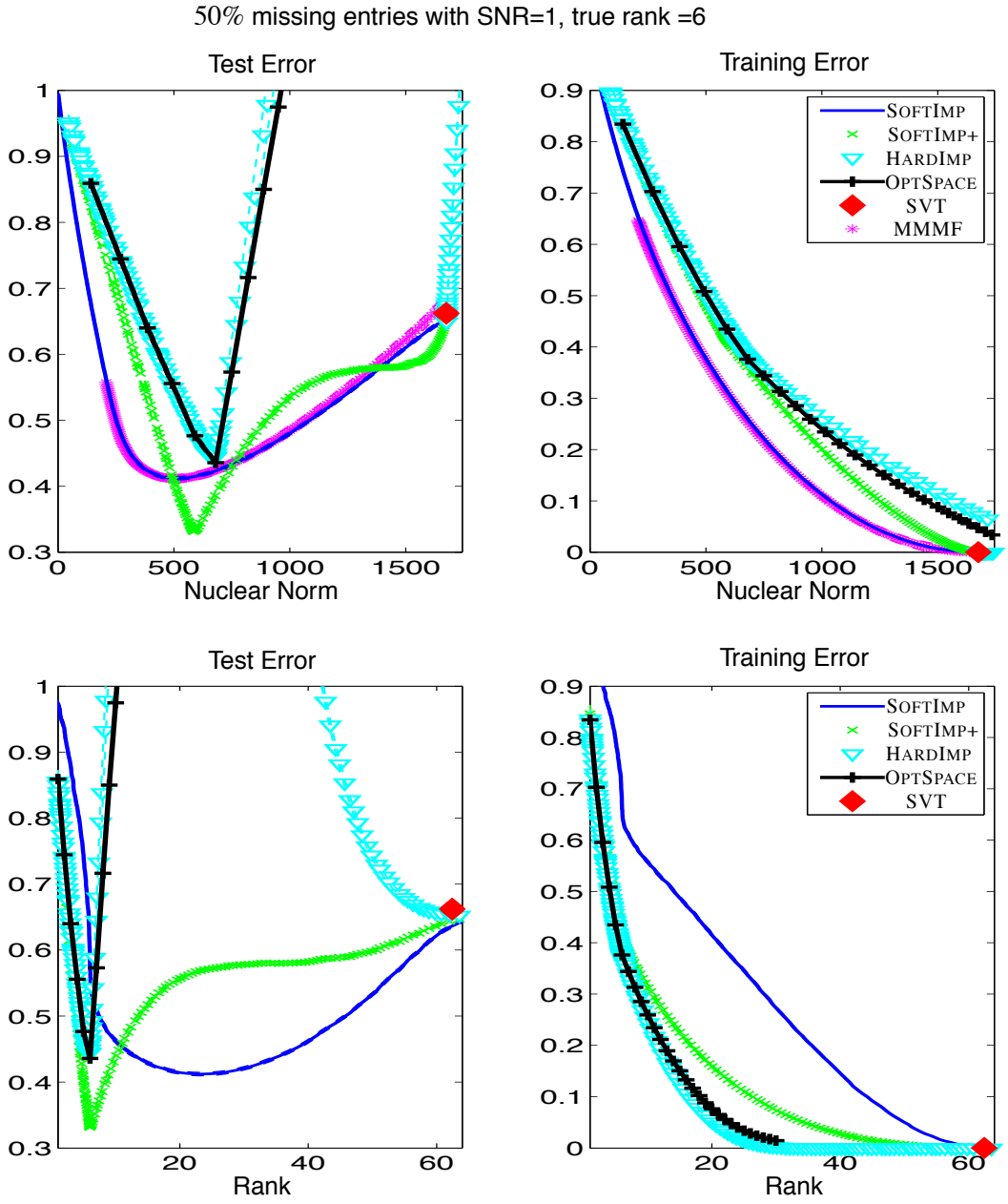


Figure 3: SOFT-IMPUTE+ has the best prediction error, closely followed by SOFT-IMPUTE and MMMF. Both HARD-IMPUTE and OPTSPACE have poor prediction error apart from near the true rank 6 of the matrix, where they show reasonable performance. SVT has very poor prediction error; it recovers a matrix with high nuclear norm and rank > 60, where the true rank is only 6. OPTSPACE is evaluated for a series of ranks ≤ 35.

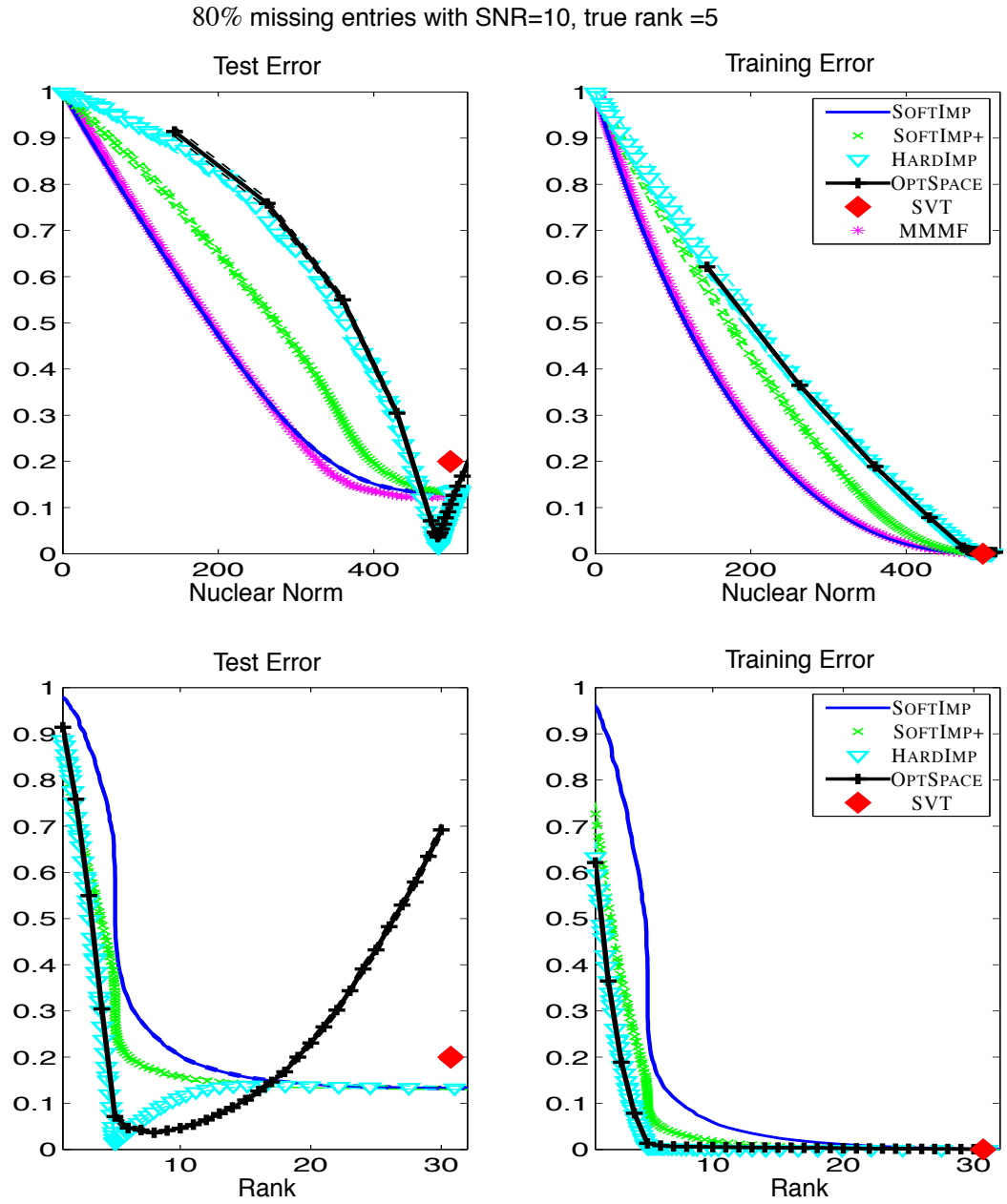


Figure 4: With low noise the performance of HARD-IMPUTE improves. It gets the correct rank whereas OPTSPACE slightly overestimates the rank. HARD-IMPUTE has the best prediction error, followed by OPTSPACE. Here MMMF has slightly better prediction error than SOFT-IMPUTE. Although the noise is low here, SVT recovers a matrix with high rank (approximately 30) and has poor prediction error as well. The test error of SVT is found to be different from the limiting solution of SOFT-IMPUTE; although in theory the limiting solution of (10) should coincide with that of SVT, in practice we never go to the limit.

In Figure 4 with SNR= 10 the noise is relatively small compared to the other two cases. The true underlying rank is 5, but the proportion of missing entries is much higher at eighty percent. Test errors of both SOFT-IMPUTE+ and SOFT-IMPUTE are found to decrease till a large nuclear norm after which they become roughly the same, suggesting no further impact of regularization. MMMF has slightly better test error than SOFT-IMPUTE around a nuclear norm of 350, while in theory they should be identical. Notice, however, that the training error is slightly worse (everywhere), suggesting that MMMF is sometimes trapped in local minima. The fact that this slightly underfit solution does better in test error is a quirk of this particular example. OPTSPACE performs well in this high-SNR example, achieving a sharp minima at the true rank of the matrix. HARD-IMPUTE performs the best in this example. The better performance of both OPTSPACE and HARD-IMPUTE over SOFT-IMPUTE can be attributed both to the low-rank truth and the high SNR. This is reminiscent of the better predictive performance of best-subset or concave penalized regression often seen over LASSO in setups where the underlying model is very sparse (Friedman, 2008).

9.2 Comparison with Fast MMMF (Rennie and Srebro, 2005)

In this section we compare SOFT-IMPUTE with MMMF in terms of computational efficiency. We also examine the consequences of two regularization parameters (r', λ) for MMMF over one for SOFT-IMPUTE.

Rennie and Srebro (2005) describes a fast algorithm based on conjugate-gradient descent for minimization of the MMMF criterion (6). With (6) being non-convex, it is hard to provide theoretical optimality guarantees for the algorithm for arbitrary r', λ—that is, what type of solution it converges to or how far it is from the global minimizer.

In Table 1 we summarize the performance results of the two algorithms. For both SOFT-IMPUTE and MMMF we consider a equi-spaced grid of 150 $\lambda \in [\lambda_{\min}, \lambda_{\max}]$, with $\lambda_{\min}$ corresponding to a full-rank solution of SOFT-IMPUTE and $\lambda_{\max}$ the zero solution. For MMMF, three different values of r' were used, and for each $(\hat{U}, \hat{V})$ were solved for over the grid of λ values. A separate held-out validation set with twenty percent of the missing entries sampled from $\Omega^{\perp}$ were used to train the tuning parameter λ (for each value of r') for MMMF and SOFT-IMPUTE. Finally we evaluate the standardized prediction errors on a test set consisting of the remaining eighty percent of the missing entries in $\Omega^{\perp}$. In all cases we report the training errors and test errors on the optimally tuned λ. SOFT-IMPUTE was run till a tolerance of 10^{-4} was achieved (fraction of decrease of objective value). Likewise for MMMF we set the tolerance of the conjugate gradient method to 10^{-4}.

In Table 1, for every algorithm total time indicates the time required for evaluating solutions over the entire grid of λ values. In these examples, we used direct SVD factorization based methods for the SVD computation, since the size of the problems were quite small. In all these examples we observe that SOFT-IMPUTE performs very favorably in terms of total times. For MMMF the time to train the models increase with increasing rank r'; and in case the underlying matrix has rank which is larger than r', the computational cost will be large in order to get competitive predictive accuracy. This point is supported in the examples of Table 1. It is important to note that, the prediction error of SOFT-IMPUTE as obtained on the validation set is actually within standard error of the best prediction error produced by all the MMMF models. In addition we also performed some medium-scale examples increasing the dimensions of the matrices. To make comparisons fair, SOFT-IMPUTE made use of *direct* SVD computations (in MATLAB) instead of iterative algorithms

Data	Method (rank)	Test error	Training error	Time (secs)
$(m,n) = (10^2, 10^2)$	SOFT-IMPUTE (39)	0.7238 (0.0027)	0.0720	4.1745
$\|\Omega\| = 5 \times 10^3 (50\%)$	MMMF (20)	0.7318 (0.0031)	0.2875	48.1090
SNR= 3	MMMF (60)	0.7236 (0.0027)	0.1730	62.0230
rank (R)= 30	MMMF (100)	0.7237 (0.0027)	0.1784	96.2750
$(m,n) = (10^2, 10^2)$	SOFT-IMPUTE (37)	0.5877 (0.0047)	0.0017	4.0976
$\|\Omega\| = 2 \times 10^3 (20\%)$	MMMF (20)	0.5807 (0.0049)	0.0186	53.7533
SNR= 10	MMMF (60)	0.5823 (0.0049)	0.0197	62.0230
rank(R)= 10	MMMF (100)	0.5823 (0.0049)	0.0197	84.0375
$(m,n) = (10^2, 10^2)$	SOFT-IMPUTE (59)	0.6008 (0.0028)	0.0086	3.8447
$\|\Omega\| = 8 \times 10^3 (80\%)$	MMMF (20)	0.6880 (0.0029)	0.4037	33.8685
SNR= 10	MMMF (60)	0.5999 (0.0028)	0.0275	57.3488
rank(R)= 45	MMMF (100)	0.5999 (0.0028)	0.0275	89.4525

Table 1: Performances of SOFT-IMPUTE and MMMF for different problem instances, in terms of test error (with standard errors in parentheses), training error and times for learning the models. SOFT-IMPUTE,"rank" denotes the rank of the recovered matrix, at the optimally chosen value of λ. For the MMMF, "rank" indicates the value of r' in $U_{m\times r'}, V_{n\times r'}$. Results are averaged over 50 simulations.

exploiting the specialized *Sparse+Low-Rank* structure (22). We report our findings on one such simulation example:

- For $(m,n) = (2000, 1000)$, $|\Omega|/(m \cdot n) = 0.2$, rank $= 500$ and SNR=10; SOFT-IMPUTE takes 1.29 hours to compute solutions on a grid of 100 λ values. The test error on the validation set and training error are 0.9630 and 0.4375 with the recovered solution having a rank of 225.

  For the same problem, MMMF with $r' = 200$ takes 6.67 hours returning a solution with test-error 0.9678 and training error 0.6624. With $r' = 400$ it takes 12.89 hrs with test and training errors 0.9659 and 0.6564 respectively.

We will like to note that DeCoste (2006) proposed an efficient implementation of MMMF via an ensemble based approach, which is quite different in spirit from the batch optimization algorithms we are studying in this paper. Hence we do not compare it with SOFT-IMPUTE.

9.3 Comparison with Nesterov's Accelerated Gradient Method

Ji and Ye (2009) proposed a first-order algorithm based on Nesterov's acceleration scheme (Nesterov, 2007), for nuclear norm minimization for a generic multi-task learning problem (Argyriou et al., 2008, 2007). Their algorithm (Liu et al., 2009; Ji and Ye, 2009) can be adapted to the SOFT-IMPUTE problem (10); hereafter we refer to it as NESTEROV. It requires one to compute the SVD of a dense matrix having the dimensions of X, which makes it prohibitive for large-scale problems. We instead would make use of the structure (22) for the SVD computation, a special characteristic of matrix completion which is not present in a generic multi-task learning problem. Here we compare the performances of SOFT-IMPUTE and NESTEROV on small scale examples, where direct SVDs can be computed easily.

Since both algorithms solve the same criterion, the quality of the solutions—objective values, training and test errors—will be the same (within tolerance). We hence compare their performances based on the times taken by the algorithms to converge to the optimal solution of (10) on a grid of values of λ. Both algorithms compute a path of solutions using warm starts. Results are shown in Figure 5, for four different scenarios described in Table 2.

Example	(m,n)	$\|\Omega\|/(m\cdot n)$	Rank	Test Error
i	$(100,100)$	0.5	5	0.4757
ii	$(100,100)$	0.2	5	0.0668
iii	$(100,100)$	0.8	5	0.0022
iv	$(1000,500)$	0.5	30	0.0028

Table 2: Four different examples used for timing comparisons of SOFT-IMPUTE and NESTEROV (accelerated Nesterov algorithm of Ji and Ye 2009). In all cases the SNR= 10.

Figure 5 shows the time to convergence for the two algorithms. Their respective number of iterations are not comparable. This is because NESTEROV uses a line-search to compute an adaptive step-size (approximate the Lipschitz constant) at every iteration, whereas SOFT-IMPUTE does not.

SOFT-IMPUTE has a rate of convergence given by Theorem 2, which for large k is worse than the accelerated version NESTEROV with rate $O(1/k^2)$. However, timing comparisons in Figure 5 show that SOFT-IMPUTE performs very favorably. We do not know the exact reason behind this, but mention some possibilities. Firstly the rates are *worst case* convergence rates. On particular problem instances of the form (10), the rates of convergence in *practice* of SOFT-IMPUTE and NESTEROV may be quite similar. Since Ji and Ye (2009) uses an adaptive step-size strategy, the choice of a step-size may be time consuming. SOFT-IMPUTE on the other hand, uses a constant step size.

Additionally, it appears that the use of the *momentum* term in NESTEROV affects the *Sparse+Low-rank* decomposition (22). This may prevent the algorithm to be adapted for solving large problems, due to costly SVD computations.

9.4 Large Scale Simulations for SOFT-IMPUTE

Table 3 reports the performance of SOFT-IMPUTE on some large-scale problems. All computations are performed in MATLAB and the MATLAB implementation of PROPACK is used. Data input, access and transfer in MATLAB take a sizable portion of the total computational time, given the size of these problems. However, the main computational bottle neck in our algorithm is the structured SVD computation. In order to focus more on the essential computational task, Table 3 displays the total time required to perform the SVD computations over all iterations of the algorithm. Note that for all the examples considered in Table 3, the implementations of algorithms NESTEROV (Liu et al., 2009; Ji and Ye, 2009) and MMMF (Rennie and Srebro, 2005) are prohibitively expensive both in terms of computational time and memory requirements, and hence could not be run. We used the value $\lambda = ||P_\Omega(X)||_2/K$ with SOFT-IMPUTE, with $K = 1.5$ for all examples but the last, where $K = 2$. $\lambda_0 = ||P_\Omega(X)||_2$ is the largest singular value of the input matrix X (padded with zeros); this is the smallest value of λ for which $\mathbf{S}_{\lambda_0}(P_\Omega(X)) = 0$ in the first iteration of SOFT-IMPUTE (Section 3).

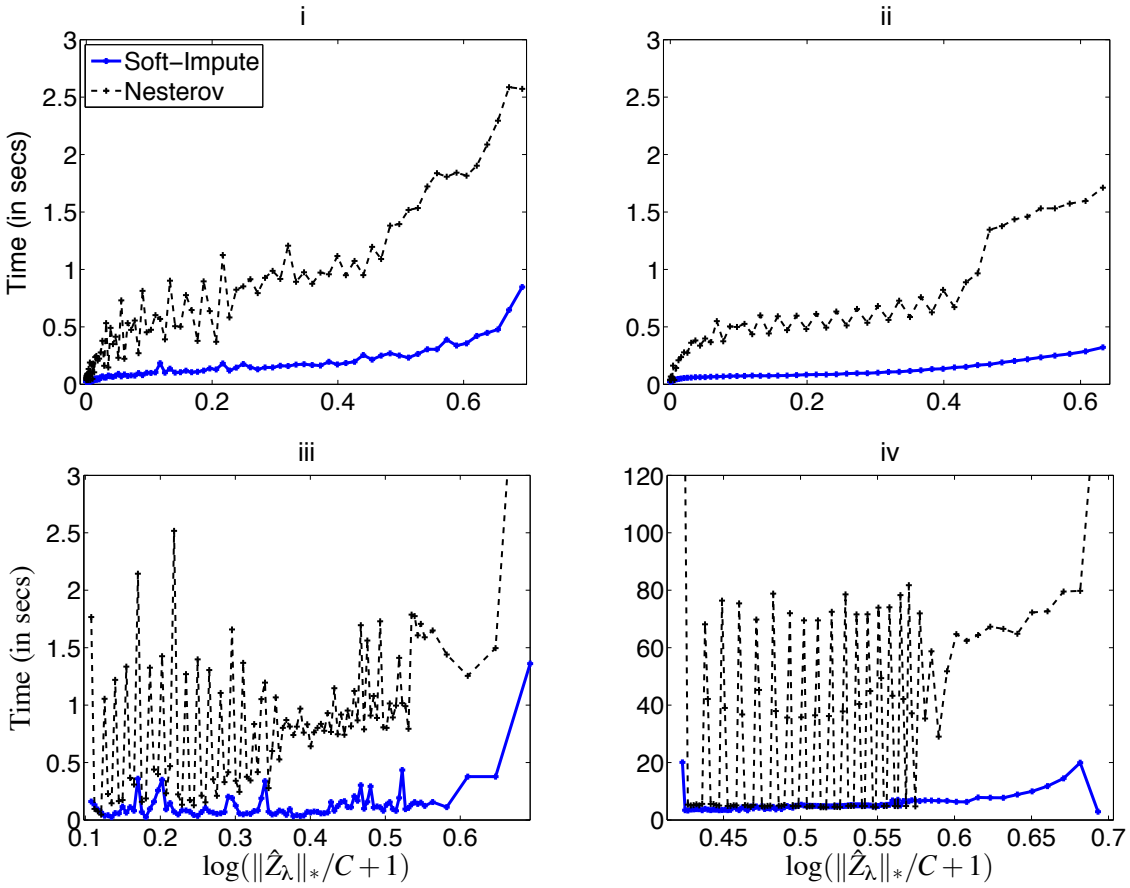


Figure 5: Timing comparisons of SOFT-IMPUTE and NESTEROV (accelerated Nesterov algorithm of Ji and Ye 2009). The horizontal axis corresponds to the standardized nuclear norm, with $C = \max_\lambda \|\hat{Z}_\lambda\|_*$. Shown are the times till convergence for the two algorithms over an entire grid of λ values for examples i–iv (in the last the matrix dimensions are much larger). The overall time differences between Examples i–iii and Example iv is due to the increased cost of the SVD computations. Results are averaged over 10 simulations. The times for NESTEROV change far more erratically with λ than they do for SOFT-IMPUTE.

The prediction performance is awful for all but one of the models, because in most cases the fraction of observed data is very small. These simulations were mainly to show the computational capabilities of SOFT-IMPUTE on very large problems.

10. Application to the Netflix Data Set

The Netflix training data consists of the ratings of 17,770 movies by 480,189 Netflix customers. The resulting data matrix is extremely sparse, with 100,480,507 or 1% of the entries observed. The task was to predict the unseen ratings for a qualifying set and a test set of about 1.4 million ratings each, with the true ratings in these data sets held in secret by Netflix. A probe set of about 1.4 million

(m,n)	$\|\Omega\|$	$\|\Omega\|/(m\cdot n)$ $\times 100\%$	Recovered rank	Time (mins)	Test error	Training error
$(10^4,10^4)$	10^5	0.1	40^*	0.2754	0.9946	0.6160
$(10^4,10^4)$	10^5	0.1	40^*	0.3770	0.9959	0.6217
$(10^4,10^4)$	10^5	0.1	50^*	0.4292	0.9962	0.3862
$(10^4,10^4)$	10^6	1.0	5	1.6664	0.6930	0.6600
$(10^5,10^5)$	5×10^6	0.05	40^*	17.2518	0.9887	0.8156
$(10^5,10^5)$	10^7	0.1	5	20.3142	0.9803	0.9761
$(10^6,10^5)$	10^8	0.1	5	259.9620	0.9913	0.9901
$(10^6,10^6)$	10^7	0.001	20^*	99.6780	0.9998	0.5834

Table 3: Performance of SOFT-IMPUTE on different problem instances. All models are generated with SNR=10 and underlying rank=5. Recovered rank is the rank of the solution matrix $\hat{Z}$ at the value of λ used in (10). Those with stars reached the "maximum rank" threshold, and option in our algorithm. Convergence criterion is taken as "fraction of improvement of objective value" less than 10^{-4} or a maximum of 15 iterations for the last four examples. All implementations are done in MATLAB including the MATLAB implementation of PROPACK on a Intel Xeon Linux 3GHz processor.

ratings was distributed to participants, for calibration purposes. The movies and customers in the qualifying, test and probe sets are all subsets of those in the training set.

The ratings are integers from 1 (poor) to 5 (best). Netflix's own algorithm has an RMSE of 0.9525, and the contest goal was to improve this by 10%, or an RMSE of 0.8572 or better. The contest ran for about 3 years, and the winning team was "Bellkor's Pragmatic Chaos", a merger of three earlier teams (see `http://www.netflixprize.com/` for details). They claimed the grand prize of $1M on September 21, 2009.

Many of the competitive algorithms build on a regularized low-rank factor model similar to (6) using randomization schemes like mini-batch, stochastic gradient descent or sub-sampling to reduce the computational cost over making several passes over the entire data-set (see Salakhutdinov et al., 2007; Bell and Koren., 2007; Takacs et al., 2009, for example). In this paper, our focus is not on using randomized or sub-sampling schemes. Here we demonstrate that our nuclear-norm regularization algorithm can be applied in batch mode on the entire Netflix training set with a reasonable computation time. We note however that the conditions under which the nuclear-norm regularization is theoretically meaningful (Candès and Tao, 2009; Srebro et al., 2005a) are not met on the Netflix data set.

We applied SOFT-IMPUTE to the training data matrix and then performed a least-squares unshrinking on the singular values with the singular vectors and the training data row and column means as the bases. The latter was performed on a data-set of size 10^5 randomly drawn from the probe set. The prediction error (RMSE) is obtained on a left out portion of the probe set. Table 4 reports the performance of the procedure for different choices of the tuning parameter λ (and the corresponding rank); times indicate the total time taken for the SVD computations over all iterations. A maximum of 10 iterations were performed for each of the examples. Again, these results are not competitive with those of the competition leaders, but rather demonstrate the feasibility of applying SOFT-IMPUTE to such a large data set.

λ	Rank	Time (hrs)	RMSE
$\lambda_0/250$	42	1.36	0.9622
$\lambda_0/300$	66	2.21	0.9572
$\lambda_0/500$	81	2.83	0.9543
$\lambda_0/600$	95	3.27	0.9497

Table 4: Results of applying SOFT-IMPUTE to the Netflix data. $\lambda_0 = ||P_\Omega(X)||_2$; see Section 9.4. The computations were done on a Intel Xeon Linux 3GHz processor; timings are reported based on MATLAB implementations of PROPACK and our algorithm. RMSE is root-mean squared error, as defined in the text.

Acknowledgments

We thank the reviewers for their suggestions that lead to improvements in this paper. We thank Stephen Boyd, Emmanuel Candes, Andrea Montanari, and Nathan Srebro for helpful discussions. Trevor Hastie was partially supported by grant DMS-0505676 from the National Science Foundation, and grant 2R01 CA 72028-07 from the National Institutes of Health. Robert Tibshirani was partially supported from National Science Foundation Grant DMS-9971405 and National Institutes of Health Contract N01-HV-28183.

Appendix A. Proofs

We begin with the proof of Lemma 1.

A.1 Proof of Lemma 1

Proof Let $Z = \tilde{U}_{m\times n}\tilde{D}_{n\times n}\tilde{V}'_{n\times n}$ be the SVD of Z. Assume without loss of generality, $m \geq n$. We will explicitly evaluate the closed form solution of the problem (8). Note that

$$\frac{1}{2}\|Z - W\|_F^2 + \lambda\|Z\|_* = \frac{1}{2}\left\{\|Z\|_F^2 - 2\sum_{i=1}^n \tilde{d}_i\tilde{u}_i' W\tilde{v}_i + \sum_{i=1}^n \tilde{d}_i^2\right\} + \lambda\sum_{i=1}^n \tilde{d}_i \tag{32}$$

where

$$\tilde{D} = \text{diag}\left[\tilde{d}_1,\ldots,\tilde{d}_n\right],\ \tilde{U} = [\tilde{u}_1,\ldots,\tilde{u}_n],\ \tilde{V} = [\tilde{v}_1,\ldots,\tilde{v}_n].$$

Minimizing (32) is equivalent to minimizing

$$-2\sum_{i=1}^n \tilde{d}_i\tilde{u}_i' W\tilde{v}_i + \sum_{i=1}^n \tilde{d}_i^2 + \sum_{i=1}^n 2\lambda\tilde{d}_i;\ \text{ w.r.t. } (\tilde{u}_i,\tilde{v}_i,\tilde{d}_i),\ i = 1,\ldots,n,$$

under the constraints $\tilde{U}'\tilde{U} = I_n$, $\tilde{V}'\tilde{V} = I_n$ and $\tilde{d}_i \geq 0\ \ \forall i$.

Observe the above is equivalent to minimizing (w.r.t. $\tilde{U},\tilde{V}$) the function $Q(\tilde{U},\tilde{V})$:

$$Q(\tilde{U},\tilde{V}) = \min_{\tilde{D}\geq 0}\quad \frac{1}{2}\left\{-2\sum_{i=1}^n \tilde{d}_i\tilde{u}_i' W\tilde{v}_i + \sum_{i=1}^n \tilde{d}_i^2\right\} + \lambda\sum_{i=1}^n \tilde{d}_i. \tag{33}$$

Since the objective (33) to be minimized w.r.t. $\tilde{D}$, is separable in $\tilde{d}_i, i = 1,\ldots,n$; it suffices to minimize it w.r.t. each $\tilde{d}_i$ separately.

The problem

$$\underset{\tilde{d}_i \geq 0}{\text{minimize}} \frac{1}{2}\left\{-2\tilde{d}_i\tilde{u}_i' W\tilde{v}_i + \tilde{d}_i^2\right\} + \lambda\tilde{d}_i \tag{34}$$

can be solved looking at the stationary conditions of the function using its sub-gradient (Nesterov, 2003). The solution of the above problem is given by $S_\lambda(\tilde{u}_i' W\tilde{v}_i) = (\tilde{u}_i' W\tilde{v}_i - \lambda)_+$, the soft-thresholding of $\tilde{u}_i' W\tilde{v}_i$ (without loss of generality, we can take $\tilde{u}_i' W\tilde{v}_i$ to be non-negative). More generally the soft-thresholding operator (Friedman et al., 2007; Hastie et al., 2009) is given by $S_\lambda(x) = \text{sgn}(x)(|x| - \lambda)_+$. See Friedman et al. (2007) for more elaborate discussions on how the soft-thresholding operator arises in univariate penalized least-squares problems with the ℓ_1 penalization.

Plugging the values of optimal $\tilde{d}_i, i = 1,\ldots,n$; obtained from (34) in (33) we get

$$Q(\tilde{U},\tilde{V}) = \frac{1}{2}\left\{\|Z\|_F^2 - 2\sum_{i=1}^{n}(\tilde{u}_i' W\tilde{v}_i - \lambda)_+(\tilde{u}_i' W\tilde{v}_i - \lambda) + (\tilde{u}_i' X\tilde{v}_i - \lambda)_+^2\right\}. \tag{35}$$

Minimizing $Q(\tilde{U},\tilde{V})$ w.r.t. $(\tilde{U},\tilde{V})$ is equivalent to maximizing

$$\sum_{i=1}^{n}\left\{2(\tilde{u}_i' W\tilde{v}_i - \lambda)_+(\tilde{u}_i' W\tilde{v}_i - \lambda) - (\tilde{u}_i' W\tilde{v}_i - \lambda)_+^2\right\} = \sum_{\tilde{u}_i' W\tilde{v}_i > \lambda}(\tilde{u}_i' W\tilde{v}_i - \lambda)^2. \tag{36}$$

It is a standard fact that for every i the problem

$$\underset{\|u\|_2^2 \leq 1, \|v\|_2^2 \leq 1}{\text{maximize}} \; u' W v; \;\; \text{such that } \; u \perp \{\hat{u}_1,\ldots,\hat{u}_{i-1}\},\; v \perp \{\hat{v}_1,\ldots,\hat{v}_{i-1}\}$$

is solved by $\hat{u}_i, \hat{v}_i$, the left and right singular vectors of the matrix W corresponding to its i^{th} largest singular value. The maximum value equals the singular value. It is easy to see that maximizing the expression to the right of (36) wrt $(u_i, v_i), i = 1,\ldots,n$ is equivalent to maximizing the individual terms $\tilde{u}_i' W\tilde{v}_i$. If $r(\lambda)$ denotes the number of singular values of W larger than λ then the $(\tilde{u}_i, \tilde{v}_i), i = 1,\ldots$ that maximize the expression (36) correspond to $\left[u_1,\ldots,u_{r(\lambda)}\right]$ and $\left[v_1,\ldots,v_{r(\lambda)}\right]$; the $r(\lambda)$ left and right singular vectors of W corresponding to the largest singular values. From (34) the optimal $\tilde{D} = \text{diag}\left[\tilde{d}_1,\ldots,\tilde{d}_n\right]$ is given by $D_\lambda = \text{diag}\left[(d_1 - \lambda)_+,\ldots,(d_n - \lambda)_+\right]$.

Since the rank of W is r, the minimizer $\hat{Z}$ of (8) is given by $UD_\lambda V'$ as in (9). ■

Remark 1 *For a more general spectral regularization of the form $\lambda\sum_i \mathbf{p}(\gamma_i(Z))$ (as compared to $\sum_i \lambda\gamma_i(Z)$ used above) the optimization problem (34) will be modified accordingly. The solution of the resultant univariate minimization problem will be given by $S_\lambda^{\mathbf{p}}(\tilde{u}_i' W\tilde{v}_i)$ for some generalized "thresholding operator" $S_\lambda^{\mathbf{p}}(\cdot)$, where*

$$S_\lambda^{\mathbf{p}}(\tilde{u}_i' W\tilde{v}_i) = \underset{\tilde{d}_i \geq 0}{\arg\min}\frac{1}{2}\left\{-2\tilde{d}_i\tilde{u}_i' W\tilde{v}_i + \tilde{d}_i^2\right\} + \lambda\mathbf{p}(\tilde{d}_i).$$

The optimization problem analogous to (35) will be

$$\underset{\tilde{U},\tilde{V}}{\text{minimize}} \;\; \frac{1}{2}\left\{\|Z\|_F^2 - 2\sum_{i=1}^{n}\hat{d}_i\tilde{u}_i' W\tilde{v}_i + \sum_{i=1}^{n}\hat{d}_i^2\right\} + \lambda\sum_i \mathbf{p}(\hat{d}_i), \tag{37}$$

where $\hat{d}_i = S^{\mathbf{p}}_{\lambda}(\tilde{u}_i' W \tilde{v}_i), \forall i$. Any spectral function for which the above (37) is monotonically increasing in $\tilde{u}_i' W \tilde{v}_i$ for every i can be solved by a similar argument as given in the above proof. The solution will correspond to the first few largest left and right singular vectors of the matrix W. The optimal singular values will correspond to the relevant shrinkage/ threshold operator $S^{\mathbf{p}}_{\lambda}(\cdot)$ operated on the singular values of W. In particular for the indicator function $\mathbf{p}(t) = \lambda \mathbf{1}(t \neq 0)$, the top few singular values (un-shrunk) and the corresponding singular vectors is the solution.

A.2 Proof of Lemma 3

This proof is based on sub-gradient characterizations and is inspired by some techniques used in Cai et al. (2008).

Proof From Lemma 1, we know that if $\hat{Z}$ solves the problem (8), then it satisfies the sub-gradient stationary conditions:

$$0 \in -(W - \hat{Z}) + \lambda \partial \|\hat{Z}\|_*. \tag{38}$$

$\mathbf{S}_\lambda(W_1)$ and $\mathbf{S}_\lambda(W_2)$ solve the problem (8) with $W = W_1$ and $W = W_2$ respectively, hence (38) holds with $W = W_1, \hat{Z}_1 = \mathbf{S}_\lambda(W_1)$ and $W = W_2, \hat{Z}_2 = \mathbf{S}_\lambda(W_1)$.

The sub-gradients of the nuclear norm $\|Z\|_*$ are given by

$$\partial \|Z\|_* = \{UV' + \omega : \omega_{m \times n},\ U'\omega = 0,\ \omega V = 0,\ \|\omega\|_2 \leq 1\}, \tag{39}$$

where $Z = UDV'$ is the SVD of Z.

Let $p(\hat{Z}_i)$ denote an element in $\partial \|\hat{Z}_i\|_*$. Then

$$\hat{Z}_i - W_i + \lambda p(\hat{Z}_i) = 0,\ i = 1,2.$$

The above gives

$$(\hat{Z}_1 - \hat{Z}_2) - (W_1 - W_2) + \lambda(p(\hat{Z}_1) - p(\hat{Z}_2)) = 0, \tag{40}$$

from which we obtain

$$\langle \hat{Z}_1 - \hat{Z}_2, \hat{Z}_1 - \hat{Z}_2 \rangle - \langle W_1 - W_2, \hat{Z}_1 - \hat{Z}_2 \rangle + \lambda \langle p(\hat{Z}_1) - p(\hat{Z}_2), \hat{Z}_1 - \hat{Z}_2 \rangle = 0,$$

where $\langle a, b \rangle = \text{trace}(a'b)$.

Now observe that

$$\langle p(\hat{Z}_1) - p(\hat{Z}_2), \hat{Z}_1 - \hat{Z}_2 \rangle = \langle p(\hat{Z}_1), \hat{Z}_1 \rangle - \langle p(\hat{Z}_1), \hat{Z}_2 \rangle - \langle p(\hat{Z}_2), \hat{Z}_1 \rangle + \langle p(\hat{Z}_2), \hat{Z}_2 \rangle. \tag{41}$$

By the characterization of subgradients as in (39), we have

$$\langle p(\hat{Z}_i), \hat{Z}_i \rangle = \|\hat{Z}_i\|_* \text{ and } \|p(\hat{Z}_i)\|_2 \leq 1,\ i = 1,2.$$

This implies

$$|\langle p(\hat{Z}_i), \hat{Z}_j \rangle| \leq \|p(\hat{Z}_i)\|_2 \|\hat{Z}_j\|_* \leq \|\hat{Z}_j\|_* \text{ for } i \neq j \in \{1,2\}.$$

Using the above inequalities in (41) we obtain:

$$\begin{aligned} \langle p(\hat{Z}_1), \hat{Z}_1 \rangle + \langle p(\hat{Z}_2), \hat{Z}_2 \rangle &= \|\hat{Z}_1\|_* + \|\hat{Z}_2\|_* & (42) \\ -\langle p(\hat{Z}_1), \hat{Z}_2 \rangle - \langle p(\hat{Z}_2), \hat{Z}_1 \rangle &\geq -\|\hat{Z}_2\|_* - \|\hat{Z}_1\|_*. & (43) \end{aligned}$$

Using (42,43) we see that the r.h.s. of (41) is non-negative. Hence

$$\langle p(\hat{Z}_1) - p(\hat{Z}_2), \hat{Z}_1 - \hat{Z}_2 \rangle \geq 0.$$

Using the above in (40), we obtain:

$$\|\hat{Z}_1 - \hat{Z}_2\|_F^2 = \langle \hat{Z}_1 - \hat{Z}_2, \hat{Z}_1 - \hat{Z}_2 \rangle \leq \langle W_1 - W_2, \hat{Z}_1 - \hat{Z}_2 \rangle. \tag{44}$$

Using the Cauchy-Schwarz Inequality, $\|\hat{Z}_1 - \hat{Z}_2\|_F \|W_1 - W_2\|_F \geq \langle \hat{Z}_1 - \hat{Z}_2, W_1 - W_2 \rangle$ in (44) we get

$$\|\hat{Z}_1 - \hat{Z}_2\|_F^2 \leq \langle \hat{Z}_1 - \hat{Z}_2, W_1 - W_2 \rangle \leq \|\hat{Z}_1 - \hat{Z}_2\|_F \|W_1 - W_2\|_F$$

and in particular

$$\|\hat{Z}_1 - \hat{Z}_2\|_F^2 \leq \|\hat{Z}_1 - \hat{Z}_2\|_F \|W_1 - W_2\|_F.$$

The above further simplifies to

$$\|W_1 - W_2\|_F^2 \geq \|\hat{Z}_1 - \hat{Z}_2\|_F^2 = \|\mathbf{S}_\lambda(W_1) - \mathbf{S}_\lambda(W_2)\|_F^2.$$

■

A.3 Proof of Lemma 4

Proof We will first show (15) by observing the following inequalities:

$$\begin{aligned}
\|Z_\lambda^{k+1} - Z_\lambda^k\|_F^2 &= \|\mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k)) - \mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}))\|_F^2 \\
\text{(by Lemma 3)} \quad &\leq \| \left(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^k)\right) - \left(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1})\right) \|_F^2 \\
&= \|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^{k-1})\|_F^2 \qquad (45) \\
&\leq \|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2. \qquad (46)
\end{aligned}$$

The above implies that the sequence $\{\|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2\}$ converges (since it is decreasing and bounded below). We still require to show that $\{\|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2\}$ converges to zero.

The convergence of $\{\|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2\}$ implies that:

$$\|Z_\lambda^{k+1} - Z_\lambda^k\|_F^2 - \|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2 \to 0 \text{ as } k \to \infty.$$

The above observation along with the inequality in (45,46) gives

$$\|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^{k-1})\|_F^2 - \|Z_\lambda^k - Z_\lambda^{k-1}\|_F^2 \to 0 \implies P_\Omega(Z_\lambda^k - Z_\lambda^{k-1}) \to 0, \tag{47}$$

as $k \to \infty$.

Lemma 2 shows that the non-negative sequence $f_\lambda(Z_\lambda^k)$ is decreasing in k. So as $k \to \infty$ the sequence $f_\lambda(Z_\lambda^k)$ converges.

Furthermore from (13,14) we have

$$Q_\lambda(Z_\lambda^{k+1}|Z_\lambda^k) - Q_\lambda(Z_\lambda^{k+1}|Z_\lambda^{k+1}) \to 0 \text{ as } k \to \infty,$$

which implies that

$$\|P_\Omega^\perp(Z_\lambda^k) - P_\Omega^\perp(Z_\lambda^{k+1})\|_F^2 \to 0 \text{ as } k \to \infty.$$

The above along with (47) gives

$$Z_\lambda^k - Z_\lambda^{k-1} \to 0 \text{ as } k \to \infty.$$

This completes the proof. ■

A.4 Proof of Lemma 5

Proof The sub-gradients of the nuclear norm $\|Z\|_*$ are given by

$$\partial\|Z\|_* = \{UV' + W : W_{m\times n},\ U'W = 0,\ WV = 0,\ \|W\|_2 \le 1\}, \tag{48}$$

where $Z = UDV'$ is the SVD of Z. Since Z_λ^k minimizes $Q_\lambda(Z|Z_\lambda^{k-1})$, it satisfies:

$$0 \in -(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{k-1}) - Z_\lambda^k) + \partial\|Z_\lambda^k\|_* \ \ \forall k. \tag{49}$$

Suppose Z^* is a limit point of the sequence Z_λ^k. Then there exists a subsequence $\{n_k\} \subset \{1,2,\ldots\}$ such that $Z_\lambda^{n_k} \to Z^*$ as $k \to \infty$.

By Lemma 4 this subsequence $Z_\lambda^{n_k}$ satisfies

$$Z_\lambda^{n_k} - Z_\lambda^{n_k-1} \to 0$$

implying

$$P_\Omega^\perp(Z_\lambda^{n_k-1}) - Z_\lambda^{n_k} \longrightarrow P_\Omega^\perp(Z_\lambda^*) - Z_\lambda^* = -P_\Omega(Z^*).$$

Hence,

$$(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{n_k-1}) - Z_\lambda^{n_k}) \longrightarrow (P_\Omega(X) - P_\Omega(Z_\lambda^*)). \tag{50}$$

For every k, a sub-gradient $p(Z_\lambda^k) \in \partial\|Z_\lambda^k\|_*$ corresponds to a tuple (u_k, v_k, w_k) satisfying the properties of the set $\partial\|Z_\lambda^k\|_*$ (48).

Consider $p(Z_\lambda^{n_k})$ along the sub-sequence n_k. As $n_k \longrightarrow \infty$, $Z_\lambda^{n_k} \longrightarrow Z_\lambda^*$.

Let

$$Z_\lambda^{n_k} = u_{n_k} D_{n_k} v_{n_k}',\ \ Z^* = u_\infty D^* v_\infty'$$

denote the SVD's. The product of the singular vectors converge $u_{n_k}' v_{n_k} \to u_\infty' v_\infty$. Furthermore due to boundedness (passing on to a further subsequence if necessary) $w_{n_k} \to w_\infty$. The limit $u_\infty v_\infty' + w_\infty$ clearly satisfies the criterion of being a sub-gradient of Z^*. Hence this limit corresponds to $p(Z_\lambda^*) \in \partial\|Z_\lambda^*\|_*$.

Furthermore from (49, 50), passing on to the limits along the subsequence n_k, we have

$$\mathbf{0} \in -(P_\Omega(X) - P_\Omega(Z_\lambda^*)) + \partial\|Z_\lambda^*\|_*.$$

Hence the limit point Z_λ^* is a stationary point of $f_\lambda(Z)$.

We shall now prove (16). We know that for every n_k

$$Z_\lambda^{n_k} = \mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{n_k-1})). \tag{51}$$

From Lemma 4, we know $Z_\lambda^{n_k} - Z_\lambda^{n_k-1} \to 0$. This observation along with the continuity of $\mathbf{S}_\lambda(\cdot)$ gives

$$\mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^{n_k-1})) \to \mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^*)).$$

Thus passing over to the limits on both sides of (51) we get

$$Z_\lambda^* = \mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z_\lambda^*)),$$

therefore completing the proof. ■

A.5 Proof of Lemma 6

The proof is motivated by the principle of embedding an arbitrary matrix into a positive semidefinite matrix (Fazel, 2002). We require the following proposition, which we prove using techniques used in the same reference.

Proposition 1 *Suppose matrices $W_{m\times m}, \tilde{W}_{n\times n}, Z_{m\times n}$ satisfy the following:*

$$\begin{pmatrix} W & Z \\ Z^T & \tilde{W} \end{pmatrix} \succeq 0.$$

Then $\text{trace}(W) + \text{trace}(\tilde{W}) \geq 2\|Z\|_*$.

Proof Let $Z_{m\times n} = L_{m\times r}\Sigma_{r\times r}R_{n\times r}^T$ denote the SVD of Z, where r is the rank of the matrix Z. Observe that the trace of the product of two positive semidefinite matrices is always non-negative. Hence we have the following inequality:

$$\text{trace}\begin{pmatrix} LL^T & -LR^T \\ -RL^T & RR^T \end{pmatrix}\begin{pmatrix} W & Z \\ Z^T & \tilde{W} \end{pmatrix} \succeq 0.$$

Simplifying the above expression we get:

$$\text{trace}(LL^TW) - \text{trace}(LR^TZ^T) - \text{trace}(RL^TZ) + \text{trace}(RR^T\tilde{W}) \geq 0. \tag{52}$$

Due to the orthogonality of the columns of L, R we have the following inequalities:

$$\text{trace}(LL^TW) \leq \text{trace}(W) \text{ and } \text{trace}(RR^T\tilde{W}) \leq \text{trace}(\tilde{W}).$$

Furthermore, using the SVD of Z:

$$\text{trace}(LR^TZ^T) = \text{trace}(\Sigma) = \text{trace}(LR^TZ^T).$$

Using the above in (52), we have:

$$\text{trace}(W) + \text{trace}(\tilde{W}) \geq 2\,\text{trace}(\Sigma) = 2\,\|Z\|_*.$$

■

Proof [Proof of Lemma 6.] For the matrix Z, consider any decomposition of the form $Z = \tilde{U}_{m\times r}\tilde{V}^T_{n\times r}$ and construct the following matrix

$$\begin{pmatrix} \tilde{U}\tilde{U}^T & Z \\ Z^T & \tilde{V}\tilde{V}^T \end{pmatrix} = \begin{pmatrix} \tilde{U} \\ \tilde{V} \end{pmatrix} (\tilde{U}^T\tilde{V}^T), \tag{53}$$

which is positive semidefinite. Applying Proposition 1 to the left hand matrix in (53), we have:

$$\text{trace}(\tilde{U}\tilde{U}^T) + \text{trace}(\tilde{V}\tilde{V}^T) \geq 2\,\|Z\|_*.$$

Minimizing both sides above w.r.t. the decompositions $Z = \tilde{U}_{m\times r}\tilde{V}^T_{n\times r}$; we have

$$\min_{\tilde{U},\tilde{V};\, Z=\tilde{U}\tilde{V}^T} \left\{\text{trace}(\tilde{U}\tilde{U}^T) + \text{trace}(\tilde{V}\tilde{V}^T)\right\} \geq 2\,\|Z\|_*. \tag{54}$$

Through the SVD of Z we now show that equality is attained in (54). Suppose Z is of rank $k \leq \min(m,n)$, and denote its SVD by $Z_{m\times n} = L_{m\times k}\Sigma_{k\times k}R^T_{n\times k}$. Then for $\tilde{U} = L_{m\times k}\Sigma^{\frac{1}{2}}_{k\times k}$ and $\tilde{V} = R_{n\times k}\Sigma^{\frac{1}{2}}_{k\times k}$ the equality in (54) is attained.

Hence, we have:

$$\begin{aligned} \|Z\|_* &= \min_{\tilde{U},\tilde{V};Z=\tilde{U}\tilde{V}^T} \left\{\text{trace}(\tilde{U}\tilde{U}^T) + \text{trace}(\tilde{V}\tilde{V}^T)\right\} \\ &= \min_{\tilde{U},\tilde{V};Z=\tilde{U}_{m\times k}\tilde{V}^T_{n\times k}} \left\{\text{trace}(\tilde{U}\tilde{U}^T) + \text{trace}(\tilde{V}\tilde{V}^T)\right\}. \end{aligned}$$

Note that the minimum can also be attained for matrices with $r \geq k$ or even $r \geq \min(m,n)$; however, it suffices to consider matrices with $r = k$. Also it is easily seen that the minimum cannot be attained for any $r < k$; hence the minimal rank r for which (29) holds true is $r = k$. ■

A.6 Proof of Theorem 2

There is a close resemblance between SOFT-IMPUTE and Nesterov's gradient method (Nesterov, 2007, Section 3). However, as mentioned earlier the original motivation of our algorithm is very different.

The techniques used in this proof are adapted from Nesterov (2007).

Proof Plugging $Z^k_\lambda = \tilde{Z}$ in (11), we have

$$\begin{aligned} Q_\lambda(Z|Z^k_\lambda) &= f_\lambda(Z^k_\lambda) + \frac{1}{2}\|P^\perp_\Omega(Z^k_\lambda - Z)\|^2_F \\ &\geq f_\lambda(Z^k_\lambda). \end{aligned} \tag{55}$$

Let $Z^k_\lambda(\theta)$ denote a convex combination of the optimal solution (Z^∞_λ) and the k^{th} iterate (Z^k_λ):

$$Z^k_\lambda(\theta) = \theta Z^\infty_\lambda + (1-\theta)Z^k_\lambda. \tag{56}$$

Using the convexity of $f_\lambda(\cdot)$ we get:

$$f_\lambda(Z_\lambda^k(\theta)) \leq (1-\theta) f_\lambda(Z_\lambda^k) + \theta f_\lambda(Z_\lambda^\infty). \tag{57}$$

Expanding $Z_\lambda^k(\theta)$ using (56), and simplifying $P_\Omega^\perp(Z_\lambda^k - Z_\lambda^k(\theta))$ we have:

$$\begin{aligned} \|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^k(\theta))\|_F^2 &= \theta^2 \|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^\infty)\|_F^2 \\ &\leq \theta^2 \|Z_\lambda^k - Z_\lambda^\infty\|_F^2 \qquad (58) \\ &\leq \theta^2 \|Z_\lambda^0 - Z_\lambda^\infty\|_F^2. \qquad (59) \end{aligned}$$

Line 59 follows from (58) by observing that $\|Z_\lambda^m - Z_\lambda^\infty\|_F^2 \leq \|Z_\lambda^{m-1} - Z_\lambda^\infty\|_F^2$, $\forall m$—a consequence of the inequalities (19) and (18), established in Theorem 1.

Using (55), the value of $f_\lambda(Z)$ at the $(k+1)^{\text{th}}$ iterate satisfies the following chain of inequalities:

$$\begin{aligned} f_\lambda(Z_\lambda^{k+1}) &\leq \min_Z \left\{ f_\lambda(Z) + \frac{1}{2}\|P_\Omega^\perp(Z_\lambda^k - Z)\|_F^2 \right\} \\ &\leq \min_{\theta\in[0,1]} \left\{ f_\lambda(Z_\lambda^k(\theta)) + \frac{1}{2}\|P_\Omega^\perp(Z_\lambda^k - Z_\lambda^k(\theta))\|_F^2 \right\} \qquad (60) \\ &\leq \min_{\theta\in[0,1]} \left\{ f_\lambda(Z_\lambda^k) + \theta(f_\lambda(Z_\lambda^\infty) - f_\lambda(Z_\lambda^k)) + \frac{1}{2}\theta^2 \|Z_\lambda^0 - Z_\lambda^\infty\|_F^2 \right\}. \qquad (61) \end{aligned}$$

Line 61 follows from Line 60, by using (57) and (59).

The r.h.s. expression in (61), is minimized at $\widehat{\theta}(k+1)$ given by

$$\begin{aligned} \widehat{\theta}(k+1) &= \min\{1, \theta_k\} \in [0,1], \text{ where,} \\ \theta_k &= \frac{f_\lambda(Z_\lambda^k) - f_\lambda(Z_\lambda^\infty)}{\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2}. \end{aligned}$$

If $\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2 = 0$, then we take $\theta_k = \infty$.

Note that θ_k is a decreasing sequence. This implies that if $\theta_{k_0} \leq 1$ then $\theta_m \leq 1$ for all $m \geq k_0$.

Suppose, $\theta_0 > 1$. Then $\widehat{\theta}(1) = 1$. Hence using (61) we have:

$$f_\lambda(Z_\lambda^1) - f_\lambda(Z_\lambda^\infty) \leq \frac{1}{2}\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2 \implies \theta_1 \leq \frac{1}{2}.$$

Thus we get back to the former case.

Hence $\theta_k \leq 1$ for all $k \geq 1$.

In addition, observe the previous deductions show that, if $\theta_0 > 1$ then (20) holds true for $k = 1$.

Combining the above observations, plugging in the value of $\widehat{\theta}$ in (61) and simplifying, we get:

$$f_\lambda(Z_\lambda^{k+1}) - f_\lambda(Z_\lambda^k) \leq -\frac{(f_\lambda(Z_\lambda^k) - f_\lambda(Z_\lambda^\infty))^2}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2}. \tag{62}$$

For the sake of notational convenience, we define the sequence $\alpha_k = f_\lambda(Z_\lambda^k) - f_\lambda(Z_\lambda^\infty)$. It is easily seen that α_k is a non-negative decreasing sequence.

Using this notation in (62) we get:

$$\begin{aligned} \alpha_k &\geq \frac{\alpha_k^2}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2} + \alpha_{k+1} \\ (\text{Since } \alpha_k \downarrow) &\geq \frac{\alpha_k \alpha_{k+1}}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2} + \alpha_{k+1}. \end{aligned} \tag{63}$$

Dividing both sides of the inequality in (63), by $\alpha_k \alpha_{k+1}$ we have:

$$\alpha_{k+1}^{-1} \geq \frac{1}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2} + \alpha_k^{-1}. \tag{64}$$

Summing both sides of (64) over $1 \leq k \leq (k-1)$ we get:

$$\alpha_k^{-1} \geq \frac{k-1}{2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2} + \alpha_1^{-1}. \tag{65}$$

Since $\theta_1 \leq 1$, we observe $\alpha_1/(2\|Z_\lambda^0 - Z_\lambda^\infty\|_F^2) \leq 1/2$—using this in (65) and rearranging terms we get, the desired inequality (20)—completing the proof of the Theorem. ■

References

J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, 2009.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

F. Bach. Consistency of trace norm minimization. *Journal of Machine Learning Research*, 9: 1019–1048, 2008.

R. M. Bell and Y. Koren. Lessons from the Netflix prize challenge. Technical report, AT&T Bell Laboratories, 2007.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Burer and R. D.C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–631, 2005.

J. Cai, E. J. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion, 2008. Available at `http://www.citebase.org/abstract?id=oai:arXiv.org:0810.3286`.

E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772, 2008.

E. J. Candès and T. Tao. The power of convex relaxation: near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2009.

D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 249–256. ACM, 2006.

A. Dempster, N.. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society Series B*, 39:1–38, 1977.

D. Donoho, I. Johnstone, G. Kerkyachairan, and D. Picard. Wavelet shrinkage; asymptopia? (with discussion). *Journal of the Royal Statistical Society: Series B*, 57:201–337, 1995.

M. Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University, 2002.

J. Friedman. Fast sparse regression and classification. Technical report, Department of Statistics, Stanford University, 2008.

J. Friedman, T. Hastie, H. Hoefling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 2(1):302–332, 2007.

M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, 2009. Web page and software available at `http://stanford.edu/~boyd/cvx`.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Prediction, Inference and Data Mining (Second Edition)*. Springer Verlag, New York, 2009.

S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th International Conference on Machine Learning*, pages 457–464, 2009.

R. H. Keshavan, S. Oh, and A. Montanari. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2009.

R. M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. Technical Report DAIMI PB-357, Department of Computer Science, Aarhus University, 1998.

R.M. Larsen. Propack-software for large and sparse svd calculations, 2004. Available at `http://sun.stanford.edu/~rmunk/PROPACK`.

J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009. Available at `http://www.public.asu.edu/~jye02/Software/SLEP`.

Z. Liu and L. Vandenberghe. Interior-point method for nuclear norm approximation with application to system identfication. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1235–1256, 2009.

S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming Series A,* `forthcoming`.

R. Mazumder, J. Friedman, and T. Hastie. Sparsenet: coordinate descent with non-convex penalties. Technical report, Stanford University, 2009.

Y. Nesterov. *Introductory Lectures on Convex Optimization: Basic course*. Kluwer, Boston, 2003.

Y. Nesterov. Gradient methods for minimizing composite objective function. Technical Report 76, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, 2007.

B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization, 2007. Available at `http://www.citebase.org/abstract?id=oai:arXiv.org:0706.4138`.

J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 713–719. ACM, 2005.

R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, pages 791–798. AAAI Press, 2007.

ACM SIGKDD and Netflix. Soft modelling by latent variables: the nonlinear iterative partial least squares (NIPALS) approach. In *Proceedings of KDD Cup and Workshop*, 2007. Available at `http://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings.html`.

N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

N. Srebro, N. Alon, and T. Jaakkola. Generalization error bounds for collaborative prediction with low-rank matrices. In *Advances in Neural Information Processing Systems 17*, pages 5–27. MIT Press, 2005a.

N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005b.

G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10:623–656, 2009.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

C. H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, 38(2):894–942, 2010.

Journal of Machine Learning Research 11 (2010) 2323-2360 Submitted 10/08; Revised 5/09; Published 8/10

Efficient Heuristics for Discriminative Structure Learning of Bayesian Network Classifiers

Franz Pernkopf PERNKOPF@TUGRAZ.AT
Department of Electrical Engineering
Graz University of Technology
A-8010 Graz, Austria

Jeff A. Bilmes BILMES@EE.WASHINGTON.EDU
Department of Electrical Engineering
University of Washington
Seattle, WA 98195, USA

Editor: Russ Greiner

Abstract

We introduce a simple order-based greedy heuristic for learning discriminative structure within generative Bayesian network classifiers. We propose two methods for establishing an order of N features. They are based on the conditional mutual information and classification rate (i.e., risk), respectively. Given an ordering, we can find a discriminative structure with $O\left(N^{k+1}\right)$ score evaluations (where constant k is the tree-width of the sub-graph over the attributes). We present results on 25 data sets from the UCI repository, for phonetic classification using the TIMIT database, for a visual surface inspection task, and for two handwritten digit recognition tasks. We provide classification performance for *both* discriminative *and* generative parameter learning on *both* discriminatively *and* generatively structured networks. The discriminative structure found by our new procedures significantly outperforms generatively produced structures, and achieves a classification accuracy on par with the best discriminative (greedy) Bayesian network learning approach, but does so with a factor of $\sim$10-40 speedup. We also show that the advantages of generative discriminatively structured Bayesian network classifiers still hold in the case of missing features, a case where generative classifiers have an advantage over discriminative classifiers.

Keywords: Bayesian networks, classification, discriminative learning, structure learning, graphical model, missing feature

1. Introduction

Bayesian networks (Pearl, 1988; Cowell et al., 1999) have been widely used as a space within which to search for high performing statistical pattern classifiers. Such networks can be produced in a number of ways, and ideally the structure of such networks will be learned discriminatively. By "discriminative learning" of Bayesian network structure, we mean simply that the process of learning corresponds to optimizing an objective function that is highly representative of classification error, such as maximizing class conditional likelihood, or minimizing classification error under the 0/1-loss function or some smooth convex upper-bound surrogate (Bartlett et al., 2006).

Unfortunately, learning the structure of Bayesian networks is hard. There have been a number of negative results over the past years, showing that optimally learning various forms of constrained Bayesian networks is NP-complete even in the "generative" sense. For example, it has been shown

that learning paths (Meek, 1995), polytrees (Dasgupta, 1997), k-trees (Arnborg et al., 1987) or bounded tree-width graphs (Karger and Srebro, 2001; Srebro, 2003), and general Bayesian networks (Geiger and Heckerman, 1996) are all instances of NP-complete optimization problems. Learning the best "discriminative structure" is no less difficult, largely because the cost functions that are needed to be optimized do not in general decompose (Lauritzen, 1996), but there has as of yet not been any formal hardness results in the discriminative case.

Discriminative optimization of a Bayesian network structure for the purposes of classification does have its advantages, however. For example, the resulting networks are amenable to interpretation compared to a purely discriminative model (the structure specifies conditional independencies between variables that may indicate distinctive aspects of how best to discern between objects Bilmes et al., 2001), it is simple to work with missing features and latent variables (as we show in this paper), and to incorporate prior knowledge (see below for further details). Since discriminative learning of such networks optimizes for only one inference scenario (e.g., classification) the resulting networks might be simpler or more parsimonious than generatively derived networks, may better abide Occam's razor, and may restore some of the benefits mentioned in Vapnik (1998).

Many heuristic methods have been produced in the past to learn the structure of Bayesian network classifiers. For example, Friedman et al. (1997) introduced the tree-augmented naive (TAN) Bayes approach, where a naive Bayes (NB) classifier is augmented with edges according to various conditional mutual information criteria. Bilmes (1999, 2000) introduced the *explaining away residual* (EAR) for discriminative structure learning of dynamic Bayesian networks for speech recognition applications, which also happens to correspond to "synergy" in the neural code (Brenner et al., 2000). The EAR measure is in fact an approximation to the expected class conditional distribution, and so improving EAR is likely to decrease the KL-divergence between the true class posterior and the resultant approximate class posterior. A procedure for providing a local optimum of the EAR measure was outlined in Narasimhan and Bilmes (2005) but it may be computationally expensive. Greiner and Zhou (2002); Greiner et al. (2005) express general Bayesian networks as standard logistic regression—they optimize parameters with respect to the conditional likelihood (CL) using a conjugate gradient method. Similarly, Roos et al. (2005) provide conditions for general Bayesian networks under which the correspondence to logistic regression holds. In Grossman and Domingos (2004) the conditional log likelihood (CLL) function is used to learn a discriminative structure. The parameters are set using maximum likelihood (ML) learning. They use a greedy hill climbing search with the CLL function as a scoring measure, where at each iteration one edge is added to the structure which conforms with the restrictions of the network topology (e.g., TAN) and the acyclicity property of Bayesian networks. In a similar algorithm, the classification rate (CR)[1] has also been used for discriminative structure learning (Keogh and Pazzani, 1999; Pernkopf, 2005). The hill climbing search is terminated when there is no edge which further improves the CR. The CR is the discriminative criterion with the fewest approximations, so it is expected to perform well given sufficient data. The problem, however, is that this approach is extremely computationally expensive, as a complete re-evaluation of the training set is needed for each considered edge. Many generative structure learning algorithms have been proposed and are reviewed in Heckerman (1995), Murphy (2002), Jordan (1999) and Cooper and Herskovits (1992). Independence tests may also be used for generative structure learning using, say, mutual information (de Campos, 2006) while other recent

1. Maximizing CR is equivalent to minimizing classification error which is identical to empirical risk (Vapnik, 1998) under the 0/1-loss function. We use the CR terminology in this paper since it is somewhat more consistent with previous Bayesian network discriminative structure learning literature.

independence test work includes Gretton and Gÿorfi (2008) and Zhang et al. (2009). An experimental comparison of discriminative and generative parameter training on both discriminatively and generatively structured Bayesian network classifiers has been performed in Pernkopf and Bilmes (2005). An empirical and theoretical comparison of certain discriminative and generative classifiers (specifically logistic regression and NB) is given in Ng and Jordan (2002). It is shown that for small sample sizes the generative NB classifier can outperform the discriminative model.

This work contains the following offerings. First, a new case is made for why and when discriminatively structured generative models can be usefully used to solve multi-class classification problems.

Second, we introduce a new order-based greedy search heuristic for finding discriminative structures in generative Bayesian network classifiers that is computationally efficient and that matches the performance of the currently top-performing but computationally expensive greedy "classification rate" approach. Our resulting classifiers are restricted to TAN 1-tree and TAN 2-trees, and so our method is a form of search within a complexity-constrained model space. The approach we employ looks first for an ordering of the N features according to classification based information measures. Given the resulting ordering, the algorithm efficiently discovers high-performing discriminative network structure with no more than $O\left(N^{k+1}\right)$ score evaluations where k indicates the tree-width of the sub-graph over the attributes, and where a score evaluation can either be a mutual-information or a classification error-rate query. Our order-based structure learning is based on the observations in Buntine (1991) and the framework is similar to the K2 algorithm proposed in Cooper and Herskovits (1992). We use, however, a discriminative scoring metric and suggest approaches for establishing the variable ordering based on conditional mutual information (CMI) (Cover and Thomas, 1991) and CR.

Lastly, we provide a wide variety of empirical results on a diverse collection of data sets showing that the order-based heuristic provides comparable classification results to the best procedure - the greedy heuristic using the CR score, but our approach is computationally much cheaper. Furthermore, we empirically show that the chosen approaches for ordering the variables improve the classification performance compared to simple random orderings. We experimentally compare both discriminative and generative parameter training on *both* discriminative *and* generatively structured Bayesian network classifiers. Moreover, one of the key advantages of generative models over discriminative ones is that it is still possible to marginalize away any missing features. If it is not known at training time which features might be missing, a typical discriminative model is rendered unusable. We provide empirical results showing that discriminatively learned generative models are reasonably insensitive to such missing features and retain their advantages over generative models in such case.

The organization of the paper is as follows: In Section 2, Bayesian networks are reviewed and our notation is introduced. We briefly present the NB, TAN, and 2-tree network structures. In Section 3, a practical case is made for why discriminative structure can be desirable. The most commonly used approaches for generative and discriminative structure and parameter learning are summarized in Section 4. Section 5 introduces our order-based greedy heuristic. In Section 6, we report classification results on 25 data sets from the UCI repository (Merz et al., 1997) and from Kohavi and John (1997) using all combinations of generative/discriminative structure/parameter learning. Additionally, we present classification experiments for synthetic data, for frame- and segment-based phonetic classification using the TIMIT speech corpus (Lamel et al., 1986), for a visual surface inspection task (Pernkopf, 2004), and for handwritten digit recognition using the MNIST (LeCun

et al., 1998) and USPS data set. Last, Section 7 concludes. We note that a preliminary version of a subset of our results appeared in Pernkopf and Bilmes (2008b).

2. Bayesian Network Classifiers

A Bayesian network (BN) (Pearl, 1988; Cowell et al., 1999) $\mathcal{B} = \langle \mathcal{G}, \Theta \rangle$ is a directed acyclic graph $\mathcal{G} = (\mathbf{Z}, \mathbf{E})$ consisting of a set of nodes $\mathbf{Z}$ and a set of directed edges $\mathbf{E} = \{E_{Z_i,Z_j}, E_{Z_i,Z_k}, \ldots\}$ connecting the nodes where E_{Z_i,Z_j} is an edge directed from Z_i to Z_j. This graph represents factorization properties of the distribution of a set of random variables $\mathbf{Z} = \{Z_1, \ldots, Z_{N+1}\}$. Each variable in $\mathbf{Z}$ has values denoted by lower case letters $\{z_1, z_2, \ldots, z_{N+1}\}$. We use boldface capital letters, for example, $\mathbf{Z}$, to denote a set of random variables and correspondingly boldface lower case letters denote a set of instantiations (values). Without loss of generality, in Bayesian network classifiers the random variable Z_1 represents the class variable $C \in \{1, \ldots, |C|\}$, $|C|$ is the cardinality of C or equivalently the number of classes, $\mathbf{X}_{1:N} = \{X_1, \ldots, X_N\} = \{Z_2, \ldots, Z_{N+1}\}$ denote the set of random variables of the N attributes of the classifier. Each graph node represents a random variable, while the lack of edges in a graph specifies some conditional independence relationships. Specifically, in a Bayesian network each node is independent of its non-descendants given its parents (Lauritzen, 1996). A Bayesian network's conditional independence relationships arise due to missing parents in the graph. Moreover, conditional independence can reduce computation for exact inference on such a graph. The set of parameters which quantify the network are represented by Θ. Each node Z_j is represented as a local conditional probability distribution given its parents Z_{Π_j}. We use $\theta^j_{i|h}$ to denote a specific conditional probability table entry (assuming discrete variables), the probability that variable Z_j takes on its i^{th} value assignment given that its parents Z_{Π_j} take their h^{th} (lexicographically ordered) assignment, that is, $\theta^j_{i|h} = P_\Theta\left(Z_j = i | Z_{\Pi_j} = h\right)$. Hence, h contains the parent configuration assuming that the first element of h, that is, h_1, relates to the conditioning class and the remaining elements $h \backslash h_1$ denote the conditioning on parent attribute values. The training data consists of M independent and identically distributed samples $\mathcal{S} = \{\mathbf{z}^m\}_{m=1}^M = \left\{\left(c^m, \mathbf{x}^m_{1:N}\right)\right\}_{m=1}^M$. For most of this work, we assume a complete data set with no missing values (the exception being Section 6.6 where input features are missing at test time). The joint probability distribution of the network is determined by the local conditional probability distributions as

$$P_\Theta(\mathbf{Z}) = \prod_{j=1}^{N+1} P_\Theta\left(Z_j | Z_{\Pi_j}\right)$$

and the probability of a sample $\mathbf{z}^m$ is

$$P_\Theta(\mathbf{Z} = \mathbf{z}^m) = \prod_{j=1}^{N+1} \prod_{i=1}^{|Z_j|} \prod_h \left(\theta^j_{i|h}\right)^{u^{j,m}_{i|h}},$$

where we introduced an indicator function $u^{j,m}_{i|h}$ of the m^{th} sample

$$u^{j,m}_{i|h} = \begin{cases} 1, & \text{if } z^m_j = i \text{ and } z^m_{\Pi_j} = h \\ 0, & \text{otherwise} \end{cases}.$$

In this paper, we restrict our experiments to NB, TAN 1-tree (Friedman et al., 1997), and TAN 2-tree classifier structures (defined in the next several paragraphs). The NB network assumes that

all the attributes are conditionally independent given the class label. This means that, given C, any subset of $\mathbf{X}$ is independent of any other disjoint subset of $\mathbf{X}$. As reported in the literature (Friedman et al., 1997; Domingos and Pazzani, 1997), the performance of the NB classifier is surprisingly good even if the conditional independence assumption between attributes is unrealistic or even false in most of the data. Reasons for the utility of the NB classifier range between benefits from the bias/variance tradeoff perspective (Friedman et al., 1997) to structures that are inherently poor from a generative perspective but good from a discriminative perspective (Bilmes, 2000). The structure of the naive Bayes classifier represented as a Bayesian network is illustrated in Figure 1a.

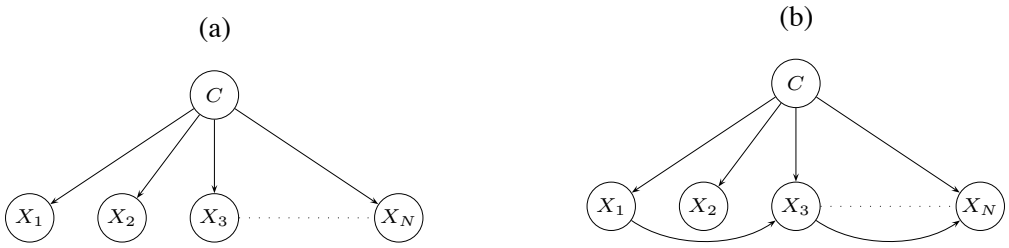


Figure 1: Bayesian Network: (a) NB, (b) TAN.

In order to correct some of the limitations of the NB classifier, Friedman et al. (1997) introduced the TAN classifier. A TAN is based on structural augmentations of the NB network, where additional edges are added between attributes in order to relax some of the most flagrant conditional independence properties of NB. Each attribute may have at most one other attribute as an additional parent which means that the tree-width of the attribute induced sub-graph is unity, that is, we have to learn a 1-tree over the attributes. The maximum number of edges added to relax the independence assumption between the attributes is $N-1$. Thus, two attributes might not be conditionally independent given the class label in a TAN. An example of a TAN 1-tree network is shown in Figure 1b. A TAN network is typically initialized as a NB network and additional edges between attributes are determined through structure learning. An extension of the TAN network is to use a k-tree, that is, each attribute can have a maximum of k attribute nodes as parents. In this work, TAN and k-tree structures are restricted such that the class node remains parent-less, that is, $C_\Pi = \emptyset$. While many other network topologies have been suggested in the past (and a good overview is provided in Acid et al., 2005), in this work we keep the class variable parent-less since it allows us to achieve one of our goals, which is to concentrating on generative models and their structures.

3. Discriminative Learning in Generative Models

A dichotomy exists between the two primary approaches to statistical pattern classifiers, *generative* and *discriminative* (Bishop and Lasserre, 2007; Jebara, 2001; Ng and Jordan, 2002; Bishop, 2006; Raina et al., 2004; Juang et al., 1997; Juang and Katagiri, 1992; Bahl et al., 1986). Under generative models, what is learned is a model of the joint probability of the features $\mathbf{X}$ and the corresponding class random variables C. Complexity penalized likelihood of the data is often the objective used for optimization, leading to standard maximum likelihood (ML) learning. Prediction with such a model is then performed either by using Bayes rule to form the class posterior probability or equivalently by forming class-prior penalized likelihood. Generative models have been widely studied, and are

desirable because they are amenable to interpretation (e.g., the structure of a generative Bayesian network specifies conditional independencies between variables that might have a useful high-level explanation). Additionally, they are amenable to a variety of probabilistic inference scenarios owing to the fact that they often decompose (Lauritzen, 1996)—the decomposition (or factorization) properties of a model are often crucial to their efficient computation.

Discriminative approaches, on the other hand, more directly represent aspects of the distribution that are important for classification accuracy, and there are a number of ways this can be done. For example, some approaches model only the class posterior probability (the conditional probability of the class given the features) or $p(C|\mathbf{X})$. Other approaches, such as support vector machines (SVMs) (Schölkopf and Smola, 2001; Burges, 1998) or neural networks (Bishop, 2006, 1995), directly model information about decision boundary sometimes without needing to concentrate on obtaining an accurate conditional distribution (neural networks, however, are also used to produce conditional distributions above and beyond just getting the class ranks correct Bishop, 1995). In each case, the objective function that is optimized is one whose minima occur not necessarily when the joint distribution $p(C,\mathbf{X})$ is accurate, but rather when the classification error rate on a training set is small. Discriminative models are usually restricted to one particular inference scenario, that is, the mapping from observed input features $\mathbf{X}$ to the unknown class output variable C, and not the other way around.

There are several reasons for using discriminative rather than generative classifiers, one of which is that the classification problem should be solved most simply and directly, and never via a more general problem such as the intermediate step of estimating the joint distribution (Vapnik, 1998). The superior performance of discriminative classifiers has been reported in many application domains (Ng and Jordan, 2002; Raina et al., 2004; Juang et al., 1997; Juang and Katagiri, 1992; Bahl et al., 1986).

Why then should we have an interest in generative models for discrimination? We address this question in the next several paragraphs. The distinction between generative and discriminative models becomes somewhat blurred when one considers that there are both generative and discriminative methods to learn a generative model, and within a generative model one may make a distinction between learning model structure and learning its parameters. In fact, in this paper, we make a clear distinction between learning the parameters of a generative model and learning the structure of a generative model. When using Bayesian networks to describe factorization properties of generative models, the structure of the model corresponds to the graph: fixing the graph, the parameters of the model are such that they must respect the factorization properties expressed by that graph. The structure of the model, however, can be independently learned, and different structures correspond to different families of graph (each family is spanned by the parameters respecting a particular structure). A given structure is then evaluated under a particular "best" set of parameter values, one possibility being the maximum likelihood settings. Of course, one could consider optimizing both parameters and structure simultaneously. Indeed, both structure and parameters are "parameters" of the model, and it is possible to learn the structure along with the parameters when a complexity penalty is applied that encourages sparse solutions, such as ℓ_1-regularization (Tibshirani, 1996) in linear regression and other models. We, however, find it useful to maintain this distinction between structure and parameters for the reason that parameter learning is inherently a continuous optimization procedure, while structure learning is inherently a combinatorial optimization problem. In our case, moreover, it is possible to stay within a given fixed-complexity model family—if we wish to stay within the family of say k-trees for fixed k, ℓ_1 regularization is not guaranteed to oblige.

Moreover, both parameters and structure of a generative model can be learned either generatively or discriminatively. Discriminative parameter learning of generative models, such as hidden Markov models (HMMs) has occurred for many years in the speech recognition community (Bahl et al., 1986; Ephraim et al., 1989; Ephraim and Rabiner, 1990; Juang and Katagiri, 1992; Juang et al., 1997; Heigold et al., 2008), and more recently in the machine learning community (Greiner and Zhou, 2002; Greiner et al., 2005; Roos et al., 2005; Ng and Jordan, 2002; Bishop and Lasserre, 2007; Pernkopf and Wohlmayr, 2009). Discriminative structure learning has also more recently received some attention (Bilmes, 1999, 2000; Pernkopf and Bilmes, 2005; Keogh and Pazzani, 1999; Grossman and Domingos, 2004). In fact, there are four possible cases of learning a generative model as depicted in Figure 2. Case A is when both structure and parameter learning is generative. Case B is when the structure is learned generatively, but the parameters are learned discriminatively. Case C is the mirror image of case B. Case D, potentially the most preferable case for classification, is where both the structure and parameters are discriminatively learned.

		Parameter Learning	
		Generative	Discriminative
Structure Learning	Generative	Case A	Case B
	Discriminative	Case C	Case D

Figure 2: Learning generative-model based classifiers: Cases for each possible combination of generative and discriminative learning of either the parameters or the structure of Bayesian network classifiers.

In this paper, we are particularly interested in learning the discriminative structure of a generative model. With a generative model, even discriminatively structured, some aspect of the joint distribution $p(C,\mathbf{X})$ is still being represented. Of course, a discriminatively structured generative model needs only represent that aspect of the joint distribution that is beneficial from a classification error rate perspective, and need not "generate" well (Bilmes et al., 2001). For this reason, it is likely that a discriminatively trained generative model will not need to be as complex as an accurate generatively trained model. In other words, the advantage of parsimony of a discriminative model over a generative model will likely be partially if not mostly recovered when one trains a generative model discriminatively. Moreover, there are a number of reasons why one might, in certain contexts, prefer a generative to a discriminative model including: parameter tying and domain knowledge-based hierarchical decomposition is facilitated; it is easy to work with structured data; there is less sensitivity to training data class skew; generative models can still be trained and structured discriminatively (as mentioned above); and it is easy to work with missing features by marginalizing over the unknown variables. This last point is particularly important: a discriminatively structured generative model still has the ability to go from $p(C,\mathbf{X})$ to $p(C,\mathbf{X}')$ where $\mathbf{X}'$ is a subset of the features in $\mathbf{X}$. This amounts to performing the marginalization $p(C,\mathbf{X}') = \sum_{\mathbf{X}\setminus\mathbf{X}'} p(C,\mathbf{X})$, something that can be tractable if the complexity class of $p(C,\mathbf{X})$ is limited (e.g., k-trees) and the variable order in the summation is chosen appropriately. In this work, we verify that a discriminatively structured model retains its advantages in the missing feature case (see Section 6.6). A discriminative model, however, is inherently conditional and it is not possible in general when some of the features are missing to go from $p(C|\mathbf{X})$ to $p(C|\mathbf{X}')$. This problem is also true for SVMs, logistic regression, and multi-layered perceptrons.

Learning a discriminatively structured generative model is inherently a combinatorial optimization problem on a "discriminative" objective function. This means that there is an algorithm that operates by tending to prefer structures that perform better on some measure that is related to classification error. Assuming sufficient training data, the ideal objective function is empirical risk under the 0/1-loss (what we call CR, or the average error rate over training data), which can be implicitly regularized by constraining the optimization process to consider only a limited complexity model family (e.g., k-trees for fixed k). In the case of discriminative parameter learning, CR can be used, but typically alternative continuous and differentiable cost functions, which may upper-bound CR and might be convex (Bartlett et al., 2006), are used and include conditional (log) likelihood $CLL(\mathcal{B}|\mathcal{S}) = \log \prod_{m=1}^{M} P_\Theta\left(C = c^m | \mathbf{X}_{1:N} = \mathbf{x}_{1:N}^m\right)$—this last objective function in fact corresponds to maximizing the mutual information between the class variable and the features (Bilmes, 2000), and can easily be augmented by a regularization term as well.

One may ask, given discriminative parameter learning, is discriminative structure still necessary? In the following, we present a simple synthetic example (similar to Narasimhan and Bilmes, 2005) and actual training and test results that indicate when a discriminative structure would be necessary for good classification performance in a generative model. The model consists of 3 binary valued attributes X_1, X_2, X_3 and a binary uniformly distributed class variable C. $\bar{X}_1$ denotes the negation of X_1. For both classes, X_1 is uniformly distributed and $X_2 = X_1$ with probability 0.5 and a uniformly distributed random number with probability 0.5. So we have the following probabilities for both classes:

$$X_1 := \begin{cases} 0 & \text{with probability } 0.5 \\ 1 & \text{with probability } 0.5 \end{cases}$$

$$X_2 := \begin{cases} X_1 & \text{with probability } 0.5 \\ 0 & \text{with probability } 0.25 \\ 1 & \text{with probability } 0.25 \end{cases}$$

For class 1, X_3 is determined according to the following:

$$X_3 := \begin{cases} X_1 & \text{with probability } 0.3 \\ X_2 & \text{with probability } 0.5 \\ 0 & \text{with probability } 0.1 \\ 1 & \text{with probability } 0.1 \end{cases}.$$

For class 2, X_3 is given by:

$$X_3 := \begin{cases} \bar{X}_1 & \text{with probability } 0.3 \\ X_2 & \text{with probability } 0.5 \\ 0 & \text{with probability } 0.1 \\ 1 & \text{with probability } 0.1 \end{cases}.$$

For both classes, the dependence between $X_1 - X_2$ is strong. The dependence $X_2 - X_3$ is stronger than $X_1 - X_3$, but only from a generative perspective (i.e., $I(X_2;X_3) > I(X_1;X_3)$ and $I(X_2;X_3|C) > I(X_1;X_3|C)$). Hence, if we were to use the strength of mutual information, or conditional mutual information, to choose the edge, we would choose $X_2 - X_3$. However, it is the $X_1 - X_3$ dependency that enables discrimination between the classes. Sampling from this distribution, we first learn structures using generative and discriminative methods, and then we perform parameter training

on these structures using either ML or CL (Greiner et al., 2005). For learning a generative TAN structure, we use the algorithm proposed by Friedman et al. (1997) which is based on optimizing the CMI between attributes given the class variable. For learning a discriminative structure, we apply our order-based algorithm proposed in Section 5 (we note that optimizing the EAR measure (Pernkopf and Bilmes, 2005) leads to similar results in this case).

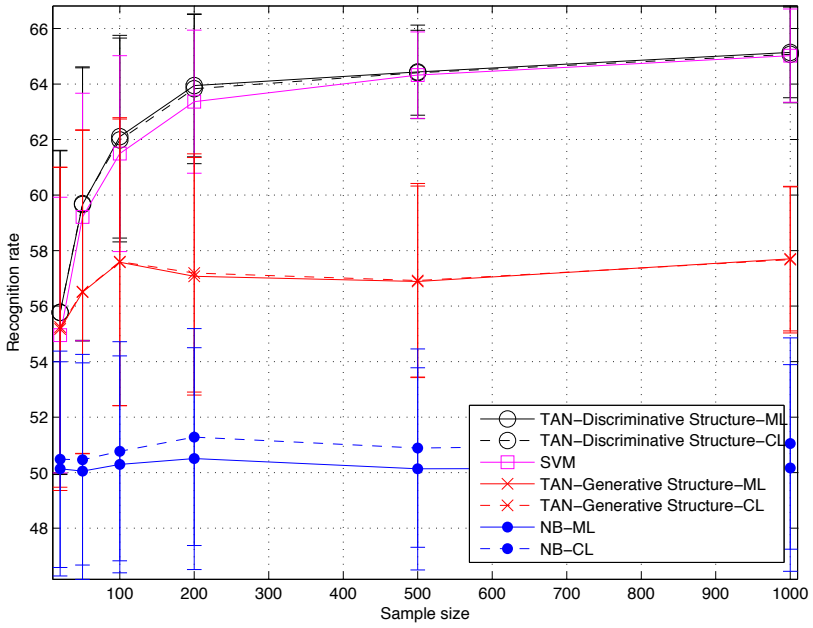


Figure 3: Generative and discriminative learning of Bayesian network classifiers on synthetic data.

Figure 3 compares the classification performance of these various cases, and in addition we show results for a NB classifier, which resorts to random guessing between both classes due to the lack of any feature dependency. Additionally, we provide the classification performance achieved with SVMs using a radial basis function (RBF) kernel.[2] On the x-axis, the training set *sample size* varies according to $\{20, 50, 100, 200, 500, 1000\}$ and the test data set contains 1000 samples. Plots are averaged over 100 independent simulations. The solid line is the performance of the classifiers using ML parameter learning, whereas, the dashed line corresponds to CL parameter training.

(a) (b)

Figure 4: (a) Generatively learned 1-tree, (b) Discriminatively learned 1-tree.

Figure 4 shows (a) the generative (b) the discriminative 1-tree over the attributes of the resulting TAN network (the class variable which is the parent of each feature is not shown in this figure). A generative model prefers edges between $X_1 - X_2$ and $X_2 - X_3$ which do not help discrimination.

2. The SVM uses two parameters C^* and σ, where C^* is the penalty parameter for the errors of the non-separable case and σ is the variance parameter for the RBF kernel. We set the values for these parameters to $C^* = 3$ and $\sigma = 1$.

The dependency between X_1 and X_3 enables discrimination to occur. Note that for this example the difference between ML and CL parameter learning is insignificant and for the generative model, only a discriminative structure enables correct classification. The performance of the non-generative SVM is similar to our discriminatively structured Bayesian network classifier. Therefore, when a generative model is desirable (see the reasons why this might be the case above), there is clearly a need for good discriminative structure learning.

In this paper, we show that the loss of a "generative meaning" of a generative model (when it is structured discriminatively) does not impair the generative model's ability to easily deal with missing features (Figure 11).

4. Learning Bayesian Networks

In the following sections, we briefly summarize state-of-the-art generative and discriminative structure and parameter learning procedures that are used to compare our order-based discriminative structure learning heuristics (which will be described in Section 5 and evaluated in Section 6).

4.1 Generative Parameter Learning

The parameters of the generative model are learned by maximizing the log likelihood of the data which leads to the ML estimation of $\theta^j_{i|h}$. The log likelihood function of a fixed structure of $\mathcal{B}$ is

$$\begin{aligned} LL(\mathcal{B}|\mathcal{S}) = \sum_{m=1}^{M} \log P_\Theta(\mathbf{Z}=\mathbf{z}^m) = \sum_{m=1}^{M}\sum_{j=1}^{N+1} \log P_\Theta\left(Z_j = z_j^m | Z_{\Pi_j} = z_{\Pi_j}^m\right) = \\ \sum_{m=1}^{M}\sum_{j=1}^{N+1}\sum_{i=1}^{|Z_j|}\sum_{h} u_{i|h}^{j,m} \log\left(\theta^j_{i|h}\right). \end{aligned} \tag{1}$$

It is easy to show that the ML estimate of the parameters is

$$\theta^j_{i|h} = \frac{\sum_{m=1}^{M} u_{i|h}^{j,m}}{\sum_{m=1}^{M}\sum_{l=1}^{|Z_j|} u_{l|h}^{j,m}},$$

using Lagrange multipliers to constrain the parameters to a valid normalized probability distribution. Since we are optimizing over constrained BN structures (k-trees), we do not perform any further regularization during training other than simple smoothing to remove zero-probability entries (see Section 6.1).

4.2 Discriminative Parameter Learning

As mentioned above, for classification purposes, having a good approximation to the posterior probability is sufficient. Hence, we want to learn parameters so that CL is maximized. Unfortunately, CL does not decompose as ML does. Consequently, there is no closed-form solution and we have to resort to iterative optimization techniques. The objective function of the conditional log likelihood

is

$$CLL(\mathcal{B}|\mathcal{S}) = \log \prod_{m=1}^{M} P_\Theta\left(C = c^m | \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right) = \sum_{m=1}^{M} \log \frac{P_\Theta\left(C = c^m, \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right)}{\sum_{c=1}^{|C|} P_\Theta\left(C = c, \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right)} =$$

$$\sum_{m=1}^{M} \left[\log P_\Theta\left(C = c^m, \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right) - \log \sum_{c=1}^{|C|} P_\Theta\left(C = c, \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right) \right].$$

Similar to Greiner and Zhou (2002) we use a conjugate gradient algorithm with line-search (Press et al., 1992). In particular, the *Polak-Ribiere* method is used (Bishop, 1995). The derivative of the objective function is

$$\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \theta^j_{i|h}} = \sum_{m=1}^{M} \left[\frac{\partial}{\partial \theta^j_{i|h}} \log P_\Theta\left(C = c^m, \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right) - \right.$$

$$\left. \frac{1}{\sum_{c=1}^{|C|} P_\Theta\left(C = c, \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right)} \frac{\partial}{\partial \theta^j_{i|h}} \sum_{c=1}^{|C|} P_\Theta\left(C = c, \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right) \right].$$

Further, we distinguish two cases for deriving $\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \theta^j_{i|h}}$. For TAN, NB, or 2-tree structures each parameter $\theta^j_{i|h}$ involves the class node value, either $C = i$ for $j = 1$ (Case A) or $C = h_1$ for $j > 1$ (Case B) where h_1 denotes the class instantiation $h_1 \in h$.

4.2.1 CASE A

For the class variable, that is, $j = 1$ and $h = \emptyset$, we get

$$\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \theta^1_i} = \sum_{m=1}^{M} \left[\frac{u^{1,m}_i}{\theta^1_i} - \frac{W^m_i}{\theta^1_i} \right],$$

where we use Equation 1 for deriving the first term (omitting the sum over j and h) and we introduced the posterior

$$W^m_i = P_\Theta\left(C = i | \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right) = \frac{P_\Theta\left(C = i, \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right)}{\sum_{c=1}^{|C|} P_\Theta\left(C = c, \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right)}.$$

4.2.2 CASE B

For the attribute variables, that is, $j > 1$, we derive correspondingly and have

$$\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \theta^j_{i|h}} = \sum_{m=1}^{M} \left[\frac{u^{j,m}_{i|h}}{\theta^j_{i|h}} - W^m_{h_1} \frac{v^{j,m}_{i|h\backslash h_1}}{\theta^j_{i|h}} \right],$$

where $W^m_{h_1} = P_\Theta\left(C = h_1 | \mathbf{X}_{1:N} = \mathbf{x}^m_{1:N}\right)$ is the posterior for class h_1 and sample m, and

$$v^{j,m}_{i|h\backslash h_1} = \begin{cases} 1, & \text{if } z^m_j = i \text{ and } z^m_{\Pi_j} = h\backslash h_1 \\ 0, & \text{otherwise} \end{cases}.$$

The probability $\theta^j_{i|h}$ is constrained to $\theta^j_{i|h} \geq 0$ and $\sum_{i=1}^{|Z_j|} \theta^j_{i|h} = 1$. We re-parameterize the problem to incorporate the constraints of $\theta^j_{i|h}$ in the conjugate gradient algorithm. Thus, we use different parameters $\beta^j_{i|h}$ as follows

$$\theta^j_{i|h} = \frac{\exp\left(\beta^j_{i|h}\right)}{\sum_{l=1}^{|Z_j|} \exp\left(\beta^j_{l|h}\right)}.$$

This requires the gradient $\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \beta^j_{i|h}}$ which is computed after some modifications as

$$\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \beta^j_{i|h}} = \sum_{k=1}^{|Z_j|} \frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \theta^j_{k|h}} \frac{\partial \theta^j_{k|h}}{\partial \beta^j_{i|h}} = \sum_{m=1}^{M} \left[u_i^{1,m} - W_i^m\right] - \theta^1_i \sum_{m=1}^{M} \sum_{c=1}^{|C|} \left[u_c^{1,m} - W_c^m\right]$$

for Case A and similarly for Case B we get the gradient

$$\frac{\partial CLL(\mathcal{B}|\mathcal{S})}{\partial \beta^j_{i|h}} = \sum_{m=1}^{M} \left[u^{j,m}_{i|h} - W^m_{h_1} v^{j,m}_{i|h\setminus h_1}\right] - \theta^j_{i|h} \sum_{m=1}^{M} \sum_{l=1}^{|Z_j|} \left[u^{j,m}_{l|h} - W^m_{h_1} v^{j,m}_{l|h\setminus h_1}\right].$$

4.3 Generative Structure Learning

The conditional mutual information between the attributes given the class variable is computed as:

$$I(X_i;X_j|C) = E_{P(X_i,X_j,C)} \log \frac{P(X_i,X_j|C)}{P(X_i|C)\,P(X_j|C)}.$$

This measures the information between X_i and X_j in the context of C. Friedman et al. (1997) gives an algorithm for constructing a TAN network using this measure. This algorithm is an extension of the approach in Chow and Liu (1968). We briefly review this algorithm in the following:

1. Compute the pairwise CMI $I(X_i;X_j|C) \quad \forall \quad 1 \leq i \leq N$ and $i < j \leq N$.
2. Build an undirected 1-tree using the maximal weighted spanning tree algorithm (Kruskal, 1956) where each edge connecting X_i and X_j is weighted by $I(X_i;X_j|C)$.
3. Transform the undirected 1-tree to a directed tree. That is, select a root variable and direct all edges away from this root. Add to this tree the class node C and the edges from C to all attributes $X_1,\ldots,X_N$.

4.4 Discriminative Structure Learning

As a baseline discriminative structure learning method, we use a greedy edge augmentation method and also the *SuperParent* algorithm (Keogh and Pazzani, 1999).

4.4.1 GREEDY HEURISTICS

While this method is expected to perform well, it is much more computationally costly then the method we propose below. The method proceeds as follows: a network is initialized to NB and at

each iteration we add the edge that, while maintaining a partial 1-tree, gives the largest improvement of the scoring function (defined below). This process is terminated when there is no edge which further improves the score. This process might thus result in a partial 1-tree (forest) over the attributes. This approach is computationally expensive since each time an edge is added, the scores for all $O\left(N^2\right)$ edges need to be re-evaluated due to the discriminative non-decomposable scoring functions we employ. This method overall has cost $O\left(N^3\right)$ score evaluations to produce a 1-tree, which in the case of an $O(NM)$) score evaluation cost (such as the below), has an overall complexity of $O\left(N^4\right)$. There are two score functions we consider: the CR (Keogh and Pazzani, 1999; Pernkopf, 2005)

$$CR\left(\mathcal{B}_{\mathcal{S}}|\mathcal{S}\right)=\frac{1}{M}\sum_{m=1}^{M}\delta\left(\mathcal{B}_{\mathcal{S}}\left(\mathbf{x}_{1:N}^{m}\right),c^{m}\right)$$

and the CL (Grossman and Domingos, 2004)

$$CL\left(\mathcal{B}|\mathcal{S}\right)=\prod_{m=1}^{M}P_{\Theta}\left(C=c^{m}|\mathbf{X}_{1:N}=\mathbf{x}_{1:N}^{m}\right),$$

where the expression $\delta\left(\mathcal{B}_{\mathcal{S}}\left(\mathbf{x}_{1:N}^{m}\right),c^{m}\right)=1$ if the Bayesian network classifier $B_{\mathcal{S}}\left(\mathbf{x}_{1:N}^{m}\right)$ trained with samples in $\mathcal{S}$ assigns the correct class label c^{m} to the attribute values $\mathbf{x}_{1:N}^{m}$, and is equal to 0 otherwise.[3] In our experiments, we consider the CR score which is directly related to the empirical risk in Vapnik (1998). The CR is the discriminative criterion that, given sufficient training data, most directly judges what we wish to optimize (error rate), while an alternative would be to use a convex upper-bound on the 0/1-loss function (Bartlett et al., 2006). Like in the generative case above, since we are optimizing over a constrained model space (k-trees), and are performing simple parameter smoothing, again regularization is implicit. This approach has in the literature been shown to be the algorithm that produces the best performing discriminative structure (Keogh and Pazzani, 1999; Pernkopf, 2005) but at the cost of a very expensive optimization procedure. To accelerate this algorithm in our implementation of this procedure (which we use as a baseline to compare against our still to-be-defined proposed approach), we apply two techniques:

1. The data samples are reordered during structure learning so that misclassified samples from previous evaluations are classified first. The classification is terminated as soon as the performance drops below the currently best network score (Pazzani, 1996).

2. During structure learning the parameters are set to the ML values. When learning the structure we only have to update the parameters of those nodes where the set of parents Z_{Π_j} changes. This observation can be also used for computing the joint probability during classification. We can memorize the joint probability and exchange only the probabilities of those nodes where the set of parents changed to get the new joint probability (Keogh and Pazzani, 1999).

In the experiments this greedy heuristic is labeled as TAN-CR and 2-tree-CR for 1-tree and 2-tree structures, respectively.

4.4.2 SUPERPARENT AND ITS k-TREE GENERALIZATION

Keogh and Pazzani (1999) introduced the *SuperParent* algorithm to efficiently learn a discriminative TAN structure. The algorithm starts with a NB network and the edges pointing from the class

3. Note that the CR scoring measure is determined from a classifier trained and tested on the same data $\mathcal{S}$.

variable to each attribute remain fixed throughout the algorithm. In the first step, each attribute in turn is considered as a parent of all other parentless attributes (except the class variable). If there are no parentless attributes left, the algorithm terminates. The parent which improves the CR the most is selected and designated the current superparent. The second step fixes the most recently chosen superparent and keeps only the single best child attribute of that superparent. The single edge between superparent and best child is then kept and the process of selecting a new superparent is repeated, unless no improvement is found at which point the algorithm terminates. The number of CR evaluations therefore in a complete run of the algorithm is $O\left(N^2\right)$. Moreover, CR determination can be accelerated as mentioned above.

We can extend this heuristic to learn 2-trees by simply modifying the first step accordingly: consider each attribute as an additional parent of all parentless or single-parented attributes (while ensuring acyclicity), and choose as the superparent the one that evaluates best, requiring $O(N)$ CR evaluations. Next, we retain the pair of edges between superparent and (parentless or single-parented) children that evaluates best using CR, requiring $O\left(N^2\right)$ CR evaluations. The process repeats if successful and otherwise terminates. The obvious k-tree generalization modifies the first step to choose an additional parent of all attributes with fewer than k parents, and then selects the best children for edge retention, leading overall to a process with $O\left(N^{k+1}\right)$ score evaluations. In this paper, we compare against this heuristic in the case of $k = 1$ and $k = 2$, abbreviating them, respectively, as *TAN-SuperParent* and *2-tree-SuperParent*.

5. New Heuristics: Order-based Greedy Algorithms

It was first noticed in Buntine (1991); Cooper and Herskovits (1992) that the best network consistent with a given variable ordering can be found with $O(N^{q+c})$ score evaluations where q is the maximum number of parents per node in a Bayesian network, and where c is a small fixed constant. These facts were recently exploited in Teyssier and Koller (2005) where generative structures were learned. Here, we are inspired by these ideas and apply them to the case of learning discriminative structures. Also, unlike Teyssier and Koller (2005), we establish only one ordering, and since our scoring cost is discriminative, it does not decompose and the learned discriminative structure is not guaranteed to be optimal. However, experiments show good results at relatively low computational learning costs.

Our procedure looks first for a total ordering $\prec$ of the variables $\mathbf{X}_{1:N}$ according to certain criteria which are outlined below. The parents of each node are chosen in such a way that the ordering is respected, and that the procedure results in at most a k-tree. We note here, a k-tree is typically defined on an undirected graphical model as one that has a tree-width of k—equivalently, there exists an elimination order in the graph such that at each elimination step, the node being eliminated has no more than k neighbors at the time of elimination. When we speak of a Bayesian network being a k-tree, what we really mean is that the moralized version of the Bayesian network is a k-tree. As a reminder, our approach is to learn a k-tree (i.e., a computationally and parameter constrained Bayesian network) over the features $\mathbf{X}_{1:N}$. We still assume, as is done with a naive Bayes model, that C is a parent of each X_i and this additional is not counted in k—thus, a 1-tree would have two parents for each X_i, both C and one additional feature. As mentioned above, in order to stay strictly within the realm of generative models, we do not consider the case where C has any parents.

5.1 Step 1: Establishing an Order $\prec$

We propose three separate heuristics for establishing an ordering $\prec$ of the attribute nodes prior to parent selection. In each case as we will see later, we use the resulting ordering such that later features may only have earlier features as parents—this limit placed on the set of parents leads to reduced computational complexity. Two of the heuristics are based on the conditional mutual information $I(C;\mathbf{X}_A|\mathbf{X}_B)$ between the class variable C and some subset of the features $\mathbf{X}_A$ given some disjoint subset of variables $\mathbf{X}_B$ (so $A \cap B = \emptyset$). The conditional mutual information (CMI) measures the degree of dependence between the class variable and $\mathbf{X}_A$ given $\mathbf{X}_B$ and it may be expressed entirely in terms of entropy via $I(C;\mathbf{X}_A|\mathbf{X}_B) = -H(C,\mathbf{X}_A,\mathbf{X}_B) + H(\mathbf{X}_A,\mathbf{X}_B) + H(C,\mathbf{X}_B) - H(\mathbf{X}_B)$, where entropy of X is defined as $H(X) = -\sum_x p(x)\log p(x)$. When $B = \emptyset$, we of course obtain (unconditional) mutual information. A structure that maximizes the mutual information between C and $\mathbf{X}$ is one that will lead to the best approximation of the posterior probability. In other words, an ideal form of optimization would do the following:

$$\mathcal{B}^* \in \operatorname*{argmax}_{\mathcal{B}\in\mathcal{F}_{\mathcal{B}}} I_{\mathcal{B}}(\mathbf{X}_{1:N};C),$$

where $\mathcal{F}_{\mathcal{B}}$ is a complexity constrained class of BNs (e.g., k-trees), and $\mathcal{B}^*$ is an optimum network. Of course, this procedure being intractable, we use mutual information to produce efficient heuristics but that we show work well in practice on a wide variety of tasks (Section 6). The third heuristic we describe is similar to the first two except that it is based directly on CR (i.e., empirical error or 0/1-loss) itself. The heuristics detailed in the following are compared against *random orderings* (RO) of the attributes in Section 6 to show that they are doing better than chance.

1: OMI: Our initial approach to finding an order is a greedy algorithm that first chooses the attribute node that is most informative about C. The next attribute in the order is the attribute node that is most informative about C conditioned on the first attribute, and subsequent nodes are chosen to be most informative about C conditioned on previously chosen attribute nodes. More specifically, our algorithm forms an ordered sequence of attribute nodes $\mathbf{X}_{\prec}^{1:N} = \left\{X_{\prec}^1, X_{\prec}^2, \ldots, X_{\prec}^N\right\}$ according to

$$X_{\prec}^j \leftarrow \operatorname*{argmax}_{X\in\mathbf{X}_{1:N}\setminus\mathbf{X}_{\prec}^{1:j-1}} \left[I\left(C;X|\mathbf{X}_{\prec}^{1:j-1}\right)\right], \tag{2}$$

where $j \in \{1,\ldots,N\}$.

It is not possible to describe the motivation for this approach without considering at least the general way parents of each attribute node are ultimately selected—more details are given below, but for now it is sufficient to say that each node's set of potential parents is restricted to come from nodes earlier in the ordering. Let $\mathbf{X}_{\Pi_j} \subseteq \mathbf{X}_{\prec}^{1:j-1}$ be the set of chosen parents for X_j in an ordering. There are several reasons why the above ordering should be useful. First, suppose we consider two potential next variables X_{j_1} and X_{j_2} as the j^{th} variable in the ordering, where $I(X_{j_1};C|\mathbf{X}_{1:j-1}) \ll I(X_{j_2};C|\mathbf{X}_{1:j-1})$. Choosing X_{j_1} could potentially lead to the case that no additional variable within the allowable set of parents $\mathbf{X}_{1:j-1}$ could be beneficially added to the model as a parent of X_{j_1}. The reason is that, conditioning on all of the potential parents of X_{j_1}, the variable X_{j_1} is less informative about C. If X_{j_2} is chosen, however, then there is a possibility that some edge augmentation as parents of X_{j_2} will render X_{j_2} residually informative about C—the reason for this is that X_{j_2} chosen to have this property, and one set of parents that renders X_{j_2} residually informative about C is the set $\mathbf{X}_{\prec}^{1:j-1}$. Stated more concisely, we wish to choose as a next variable in the ordering one that has the

potential to be strongly and residually predictive of C when choosing earlier variables as parents. When choosing X_j such that $I\left(C;X_j|\mathbf{X}_{\prec}^{1:j-1}\right)$ is large, this is possible at least in the case when X may have up to $j-1$ additional parents.

Of course, only a subset of these nodes will ultimately be chosen to ensure that the model is a k-tree and remains tractable and just because $I(X_j;C|\mathbf{X}_{1:j-1})$ is large does not necessarily mean that $I(X_j;C|\mathbf{X}_B)$ is also large for some $B \subset \{1,\ldots,(j-1)\}$. The strict sub-set relationship, where $|B| < (j-1)$, is necessary to restrict the complexity class of our models, but this goal involves an accuracy-computation tradeoff. Our approach, therefore, is only a heuristic. Nevertheless, one justification for our ordering heuristic is based on the aspect of our algorithm that achieves computational tractability, namely the parent-selection strategy where variables are only allowed to have previously ordered variables as their parents (as we describe in more detail below). Moreover, we have empirically found this property to be the case in both real and artificial random data (see below). Loosely speaking, we see our ordering as somewhat analogous to Ada-boost but applied to feature selection, where later decisions on the ordering are chosen to correct for the deficiencies of earlier decisions.

A second reason our ordering may be beneficial stems from the reason that a naive Bayes model is itself useful. In a NB, we have that each X_i is independent of X_j given C. This has beneficial properties both from the bias-variance (Friedman et al., 1997) and from the discriminative structure perspective (Bilmes, 2000). In any given ordering, variables chosen earlier in the order have more of a chance *in the resulting model* to render later variables conditionally independent of each other conditioned on both C and the earlier variable. For example, if two later variables both ask for the same earlier single parent, the two later variables are modeled as independent given C and that earlier parent. This normally would not be useful, but in our ordering, since the earlier variables are in general more correlated with C, this mimics the situation in NB: C and variables similar to C render conditionally independent other variables that are less similar to C (with NB alone, C renders all other variables conditionally independent). For reasons similar to NB (Friedman et al., 1997), such an ordering will tend to work well.

Our approach rests on being able to compute CMI queries over a large number of variables, something that requires both solving a potentially difficult inference problem and also is sensitive to training-data sparsity. In our case, however, a conditional mutual information query can be computed efficiently by making only one pass over the training data, albeit with a potential problem with bias and variance of the mutual information estimate. As mentioned above, each CMI query can be represented as a sum of signed entropy terms. Moreover, since all variables are discrete in our studies, an entropy query can be obtained in one pass over the data by computing an empirical histogram of random variable values only that exist in the data, then summing over only the resulting non-zero values. Let us assume, for simplicity, that integer variable Y represents the Cartesian product of all possible values of the vector random variable for we wish to obtain an entropy value. Normally, $H(Y) = -\sum_y p(y)\log p(y)$ would require an exponential number of terms, but we avoid this by computing $H(Y) = -\sum_{y\in\mathcal{T}_y} p(y)\log p(y) - |\mathcal{D}_y \setminus \mathcal{T}_y|\varepsilon\log\varepsilon$, where $\mathcal{T}_y$ are the set of y values that occur in the training data, and $\mathcal{D}_y$ is the set of all possible y values, and ε is any smoothing value that we might use to fill in zeros in the empirical histogram. Therefore, if our algorithm requires only a polynomial number of CMI queries, then the complexity of the algorithm is still only polynomial in the size of the training data. Of course, as the number of actual variables increases, the quality of this estimate decreases. To mitigate these problems, we can restrict the number of variables in $\mathbf{X}_{\prec}^{1:j-1}$ for a CMI query, leading to the following second heuristic based on CMI.

2: OMISP: For a 1-tree each variable $X_{\prec}^{j}$ has one single parent (SP) X_{Π_j} which is selected from the variables $\mathbf{X}_{\prec}^{1:j-1}$ appearing before $X_{\prec}^{j}$ in the ordering (i.e., $|\Pi_j| = 1, \forall j$). This leads to a simple variant of the above, where CMI conditions only on a single variable within $\mathbf{X}_{\prec}^{1:j-1}$. Under this heuristic, an ordered sequence is determined by

$$X_{\prec}^{j} \leftarrow \operatorname*{argmax}_{X \in \mathbf{X}_{1:N} \setminus \mathbf{X}_{\prec}^{1:j-1}} \left[\max_{X_{\prec} \in \mathbf{X}_{\prec}^{1:j-1}} \left[I\left(C; X | X_{\prec}\right) \right] \right].$$

Note, in this work, we do not present results using OMISP since the results were not significantly different than OMI. We refer the interested reader to Pernkopf and Bilmes (2008a) which gives the results for this heuristic, and more, in their full glory.

3: OCR: Here, CR on the training data is used in a way similar to the aforementioned greedy OMI approach. The ordered sequence of nodes $\mathbf{X}_{\prec}^{1:N}$ is determined according to

$$X_{\prec}^{j} \leftarrow \operatorname*{argmax}_{X \in \mathbf{X}_{1:N} \setminus \mathbf{X}_{\prec}^{1:j-1}} CR\left(\mathcal{B}_{\mathcal{S}} | \mathcal{S}\right),$$

where $j \in \{1, \ldots, N\}$ and the graph of $\mathcal{B}_{\mathcal{S}}$ at each evaluation is a fully connected sub-graph only over the nodes C, X, and $\mathbf{X}_{\prec}^{1:j-1}$, that is, we have a *saturated* sub-graph (note, here $\mathcal{B}_{\mathcal{S}}$ depends on the current X and the previously chosen attribute nodes in the order, but this is not indicated for notational simplicity). This can of course lead to very large local conditional probability tables. We thus perform this computation by using sparse probability tables that have been slightly smoothed as described above. We then compute CR on the basis of $P\left(C | X, \mathbf{X}_{\prec}^{1:j-1}\right) \propto P\left(X, \mathbf{X}_{\prec}^{1:j-1} | C\right) P\left(C\right)$. The justification for this approach is that it produces an ordering based not on mutual information but on a measure more directly related to classification accuracy.

5.2 Step 2: Heuristic for Selecting Parents w.r.t. a Given Order to Form a k-tree

Once we have the ordering $\mathbf{X}_{\prec}^{1:N}$, we select $X_{\Pi_j} \subseteq \mathbf{X}_{\prec}^{1:j-1}$ for each $X_{\prec}^{j}$, with $j \in \{2, \ldots, N\}$. When the size of $\mathbf{X}_{\Pi_j}$ (i.e., N) and of k are small we can use even a computational costly scoring function to find X_{Π_j}. In case of a large N, we can restrict the size of the parent set $\mathbf{X}_{\Pi_j}$ similar to the *sparse candidate algorithm* (Friedman et al., 1999). While either the CL or the CR can be used as a cost function for selecting parents, we in this work restrict our experiments to CR for parent selection (empirical results show it performed better). The parent selection proceeds as follows. For each $j \in \{2, \ldots, N\}$, we choose the best k parents $X_{\Pi_j} \subseteq \mathbf{X}_{\prec}^{1:j-1}$ for $X_{\prec}^{j}$ by scoring each of the $O\left(\binom{N}{k}\right)$ possibilities with CR. We note that for $j \in \{2, \ldots, N-1\}$ there will be a set of variables that have not yet had their parents chosen, namely variables $\mathbf{X}_{\prec}^{j+1:N}$—for these variables, we simply use the NB assumption. That is, those variables have no parents other than C for the selection of parents for $X_{\prec}^{j}$ (we relax this property in Pernkopf and Bilmes, 2008a). Note that the set of parents is judged using CR, but the model parameters for any given candidate set of parents selected are trained using ML (we did not find further advantages, in addition to using CR for parent selection, in also using discriminative parameter training). We also note that the parents for each attribute node are retained in the model only when CR is improved, and otherwise the node $X_{\prec}^{j}$ is left parent-less. This therefore might result in a partial k-tree (forest) over the attributes. We evaluate our algorithm for $k = 1$ and $k = 2$, but is defined above to learn k-trees ($k \geq 1$), and thus uses $O\left(N^{k+1}\right)$ score evaluations where,

due to ML training, each CR evaluation is $O(NM)$. Overall, for learning a 1-tree, the ordering and the parent selection costs $O\left(N^2\right)$ score evaluations. We see that the computation is comparable to that of the *SuperParent* algorithm and its k-tree generalization.

Algorithm 1 OMI-CR

Input: $\mathbf{X}_{1:N}, C, \mathcal{S}$
Output: set of edges $\mathbf{E}$ for TAN network
$X^1_\prec, X^2_\prec \leftarrow \text{argmax}_{X,X' \in \mathbf{X}_{1:N}} \left[I(C; X, X') \right]$
if $I\left(C; X^1_\prec\right) < I\left(C; X^2_\prec\right)$ **then**
  $X^2_\prec \leftrightarrow X^1_\prec$
end if
$\mathbf{E} \leftarrow \left\{ \mathbf{E}_{\text{Naive Bayes}} \cup E_{X^1_\prec, X^2_\prec} \right\}$
$j \leftarrow 2$
$CR_{old} \leftarrow 0$
repeat
  $j \leftarrow j+1$
  $X^j_\prec \leftarrow \text{argmax}_{X \in \mathbf{X}_{1:N} \setminus \mathbf{X}^{1:j-1}_\prec} \left[I\left(C; X | \mathbf{X}^{1:j-1}_\prec\right) \right]$
  $X^*_\prec \leftarrow \text{argmax}_{X \in \mathbf{X}^{1:j-1}_\prec} CR(\mathcal{B}_\mathcal{S} | \mathcal{S})$ where edges of $\mathcal{B}_\mathcal{S}$ are $\left\{ \mathbf{E} \cup E_{X, X^j_\prec} \right\}$
  $CR_{new} \leftarrow CR(\mathcal{B}_\mathcal{S} | \mathcal{S})$ where edges of $\mathcal{B}_\mathcal{S}$ are $\left\{ \mathbf{E} \cup E_{X^*_\prec, X^j_\prec} \right\}$
  if $CR_{new} > CR_{old}$ **then**
    $CR_{old} \leftarrow CR_{new}$
    $\mathbf{E} \leftarrow \left\{ \mathbf{E} \cup E_{X^*_\prec, X^j_\prec} \right\}$
  end if
until $j = N$

5.3 OMI-CR Algorithm

Recapitulating, we have introduced three order-based greedy heuristics for producing discriminative structures in Bayesian network classifiers: First, there is OMI-CR (Order based on Mutual Information with CR used for parent selection); Second, there is OMISP-CR (Order based on Mutual Information conditioned on a Single Parent, with CR used for parent selection); and third OCR-CR (Order based on Classification Rate, with CR used for parent selection). For evaluation purposes, we also consider random orderings in step 1 and CR for parent selection (RO-CR). The OMI-CR procedure is summarized in Algorithm 1 where both steps (order and parent selection) are merged at each loop iteration (which is of course equivalent to considering both steps separately). The different algorithmic variants are obtained by modifying the ordering criterion.

6. Experiments

We present classification results on 25 data sets from the UCI repository (Merz et al., 1997) and from Kohavi and John (1997), for frame- and segment-based phonetic classification experiments using the TIMIT database (Lamel et al., 1986), for a visual surface inspection task (Pernkopf, 2004), and for handwritten digit recognition using the MNIST (LeCun et al., 1998) and USPS data set.

Additionally, we show performance results on synthetic data. We use NB, TAN, and 2-tree network structures. Different combinations of the following parameter/structure learning approaches are used to learn the classifiers:

- Generative (ML) (Pearl, 1988) and discriminative (CL) (Greiner et al., 2005) parameter learning.
- CMI: Generative structure learning using CMI as proposed in Friedman et al. (1997).
- CR: Discriminative structure learning with greedy heuristic using CR as scoring function (Keogh and Pazzani, 1999; Pernkopf, 2005) (see Section 4.4).
- RO-CR: Discriminative structure learning using random ordering (RO) in step 1 and CR for parent selection in step 2 of the order-based heuristic.
- SuperParent k-tree: Discriminative structure learning using the SuperParent algorithm (Keogh and Pazzani, 1999) with $k = 1, 2$ (see Section 4.4).
- OMI-CR: Discriminative structure learning using CMI for ordering the variables (step 1) and CR for parent selection in step 2 of the order-based heuristic.
- For OMI-CR, we also evaluate discriminative parameter learning by optimizing CL during the selection of the parent in step 2. We call this OMI-CRCL. Discriminative parameter learning while optimizing discriminative structure is computationally feasible only on rather small data sets due to the cost of the conjugate gradient parameter optimization.

We do not include experimental results for OMISP-CR and OCR-CR for space reasons. The results, however, show similar performance to OMI-CR, and can be found in an extended technical-report version of this paper (Pernkopf and Bilmes, 2008a).

While we have attempted to avoid a proliferation of algorithm names, some name abundance has unavoidably occurred in this paper. We therefore have attempted to use a simple 2-, 3-, or even 4-tag naming scheme where A-B-C-D is such that "A" (if given) refers to either TAN (1-tree) or 2-tree, "B" and "C" refer to the structure learning approach, and "D" (if given) refers to the parameter training method of the *final* resultant model structure. For the ordering heuristics "B" refers to the ordering method, "C" refers to the parent selection and internal parameter learning strategy. For the remaining structure learning heuristics only "B" is present. Thus, TAN-OMI-CRML-CL would be the OMI procedure for ordering, parent selection evaluated using CR (with ML training used at that time), and with CL used to train the final model which would be a 1-tree (note moreover that TAN-OMI-CR-CL is equivalent since ML is the default training method).

6.1 Experimental Setup

Any continuous features were discretized using recursive minimal entropy partitioning (Fayyad and Irani, 1993) where the codebook is produced using only the training data. This discretization method uses the class entropy of candidate partitions to determine the bin boundaries. The candidate partition with the minimal entropy is selected. This is applied recursively on the established partitions and the minimum description length approach is used as stopping criteria for the recursive partitioning. In Dougherty et al. (1995), an empirical comparison of different discretization methods has been performed and the best results have been achieved with this entropy-based discretization. Throughout our experiments, we use exactly the same data partitioning for each training procedure. We performed simple smoothing, where zero probabilities in the conditional probability tables are replaced with small values ($\varepsilon = 0.00001$). For discriminative parameter learning, the parameters are

initialized to the values obtained by the ML approach (Greiner et al., 2005). The gradient descent parameter optimization is terminated using *cross tuning* as suggested in Greiner et al. (2005).

6.2 Data Characteristics

In the following, we introduce several data sets used in the experiments.

6.2.1 UCI DATA

We use 25 data sets from the UCI repository (Merz et al., 1997) and from Kohavi and John (1997). The same data sets, 5-fold cross-validation, and train/test learning schemes as in Friedman et al. (1997) are employed. The characteristics of the data sets are summarized in Table 7 in the Appendix A.

6.2.2 TIMIT-4/6 DATA

This data set is extracted from the TIMIT speech corpus using the dialect speaking region 4 which consists of 320 utterances from 16 male and 16 female speakers. The speech is sampled at 16 kHz. Speech frames are classified into the following classes, voiced (V), unvoiced (U), silence (S), mixed sounds (M), voiced closure (VC), and release (R) of plosives. We therefore are performing frame-by-frame phone classification (contrasted with phone *recognition* using, say, a hidden Markov model). We perform experiments with only four classes V/U/S/M and all six classes V/U/S/M/VC/R using 110134 and 121629 samples, respectively. The class distribution of the four class experiment V/U/S/M is 23.08%, 60.37%, 13.54%, 3.01% and of the six class case V/U/S/M/VC/R is 20.9%, 54.66%, 12.26%, 2.74%, 6.08%, 3.36%. Additionally, we perform classification experiments on data of male speakers (Ma), female speakers (Fe), and both genders (Ma+Fe). For each gender we have approximately the same number of samples. The data have been split into 2 mutually exclusive subsets of $\mathcal{D} \in \{\mathcal{S}_1, \mathcal{S}_2\}$ where the size of the training data $\mathcal{S}_1$ is 70% and of the test data $\mathcal{S}_2$ is 30% of $\mathcal{D}$. The classification experiments have been performed with 8 wavelet-based features combined with 12 mel-frequency cepstral coefficients (MFCC) features, that is, 20 features. More details about the features can be found in Pernkopf et al. (2008). We have 6 different classification tasks for each classifier, that is, Ma+Fe, Ma, Fe $\times$ 4 or 6 Classes.

6.2.3 TIMIT-39 DATA

This again is a phone classification test but with a larger number of classes. In accordance with Halberstadt and Glass (1997) we cluster the 61 phonetic labels into 39 classes, ignoring glottal stops. For training, 462 speakers from the standard NIST training set have been used. For testing the remaining 168 speakers from the overall 630 speakers were employed. Each speaker speaks 10 sentences including two sentences which are the same among all speakers (labeled as *sa*), five sentences which were read from a list of phonetically balanced sentences (labeled as *sx*), and 3 randomly selected sentences (labeled as *si*). In the experiments, we only use the *sx* and *si* sentences since the *sa* sentences introduce a bias for certain phonemes in a particular context. This means that we have 3696 and 1344 utterances in the training and test set, respectively. We derive from each phonetic segment one feature vector which results in 140173 training samples and 50735 testing samples. The features are derived similarly as proposed in Halberstadt and Glass (1997). First, 12 MFCC + log-energy feature (13 MFCC's) and their derivatives (13 Derivatives) are calculated for

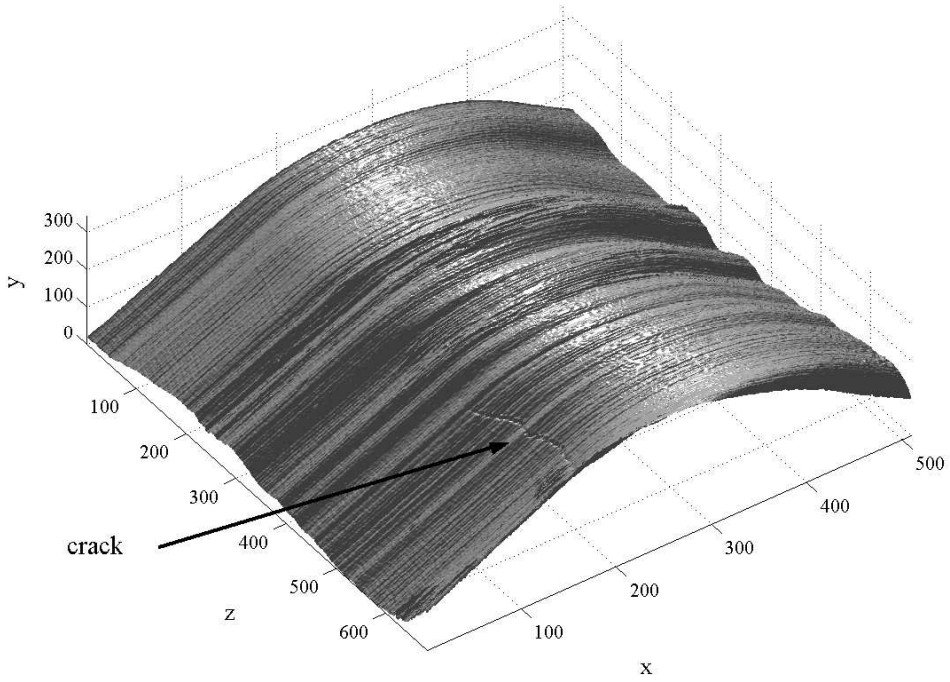


Figure 5: Acquired surface data with an embedded crack.

every 10ms of the utterance with a window size of 25ms. A phonetic segment, which can be variable length, is split at a 3:4:3 ratio into 3 parts. The fixed-length feature vector is composed of: 1) three averages of the 13 MFCC's calculated from the 3 portions (39 features); 2) the 13 Derivatives of the beginning of the first and the end of the third segment part (26 features); and 3) the log duration of the segment (1 feature). Hence, each phonetic segment is represented by 66 features.

6.2.4 SURFACE INSPECTION DATA (SURFINSP)

This data set was acquired from a surface inspection task. Surface defects with three-dimensional characteristics on scale-covered steel blocks have to be classified into 3 classes. The 3-dimensional raw data showing the case of an embedded surface crack is given in Figure 5. The data set consists of 450 surface segments uniformly distributed into three classes. Each sample (surface segment) is represented by 40 features. More details on the inspection task and the features used can be found elsewhere (Pernkopf, 2004).

6.2.5 MNIST DATA

We evaluate our classifiers on the MNIST data set of handwritten digits (LeCun et al., 1998) which contains 60000 samples for training and 10000 digits for testing. The digits are centered in a 28×28 gray-level image. We re-sample these images at a resolution of 14×14 pixels. This gives 196 features where the gray-levels are discretized using the procedure from Fayyad and Irani (1993).

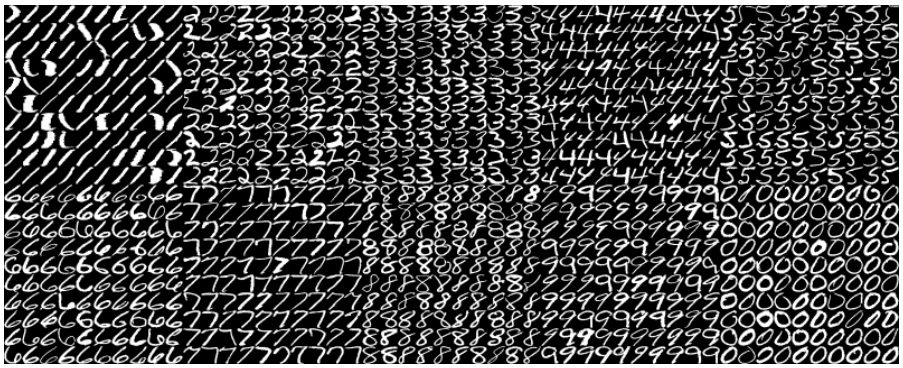


Figure 6: USPS data.

6.2.6 USPS DATA

This data set contains 11000 handwritten digit images (uniformly distributed) collected from zip codes of mail envelopes. Each digit is represented as a 16×16 grayscale image, where each pixel is considered as individual feature. Figure 6 shows a random sample of the data set. We use 8000 digits for training and the remaining images as a test set.

6.3 Conditional Likelihood and Maximum Mutual Information Orderings

In the following, we evaluate the ordering heuristics using 31 different classification scenarios (from the UCI and the TIMIT-4/6 data sets) comprising differing input features and differing numbers of classes. We compare our ordering procedure (i.e., OMI, where we maximize the mutual information as in Equation 2) with several other possible orderings in an attempt to empirically show that our aforementioned intuition regarding order (see Section 5.1) is sound in the majority of cases. In particular, we compare against an ordering produced by minimizing the mutual information (replacing argmax with argmin in Equation 2). Additionally, we also compare against 100 uniformly-at-random orderings. For the selection of the conditioning variables (see Section 5.2) the CL score is used in each case. ML parameter estimation is used for all examples in this section.

Figure 7 and Figure 8 show the resulting conditional log likelihoods (CLL) of the model scoring the training data after the TAN network structures (1-trees in this case) have been determined for the various data sets. As can be seen, our ordering heuristic performs better than both the random and the minimum mutual information orderings on 28 of the 31 cases. The random case shows the mean and $\pm$ one standard deviation out of 100 orderings. For *Corral*, *Glass*, and *Heart* there is no benefit, but the data sets are on the smaller side where it is less unexpected that generative structure learning would perform better (Ng and Jordan, 2002).

To further verify that our ordering tends to produce better conditional likelihood values on training data, we also evaluate on random data. For each number of variables (from 10 up to 14) we generate 1000 random distributions and draw 100000 samples from each one. Using these samples, we learn generative and discriminative TAN structures by the following heuristics and report the resulting conditional log likelihood on the training data: (i) order variables by maximizing mutual information (TAN-OMI-CL), (ii) order variables by minimizing mutual information, (iii) random ordering of variables and CL parent selection (TAN-RO-CL), (iv) optimal generative 1-tree, that

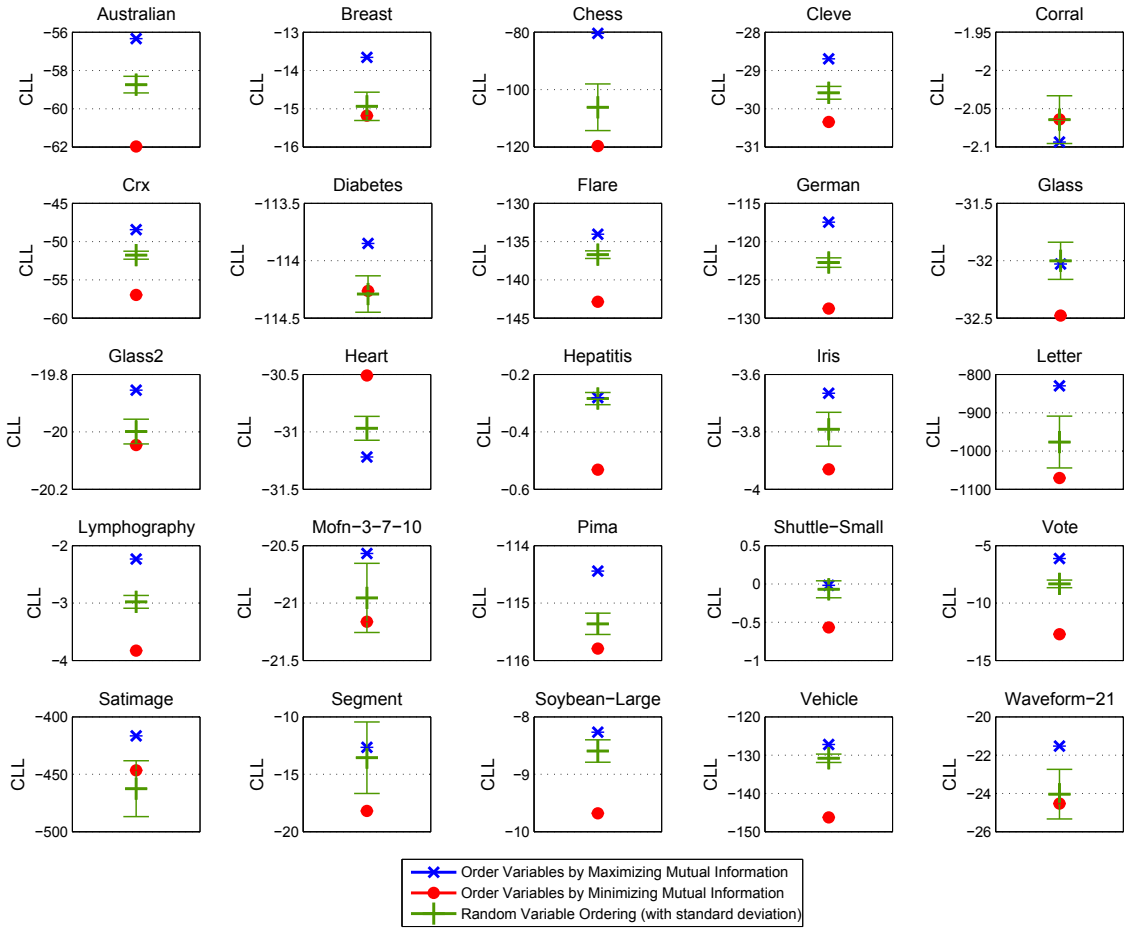


Figure 7: Resulting CLL on the UCI data sets for a maximum mutual information (i.e., OMI), a minimum mutual information, and a random based ordering scheme.

is, TAN-CMI (Friedman et al., 1997), (v) the computationally expensive greedy heuristic using CL (see Section 4.4.1), what we call TAN-CL. In addition we show CLL results for the NB classifier.

Figure 9 shows the CLL values for various algorithms. The CLL is still high even with the much less computationally costly OMI-CL procedure. Additionally, the generative 1-tree method improves likelihood but it does not necessarily produce good conditional likelihood results. We performed a one-sided paired t-test (Mitchell, 1997) for all different structure learning approaches. This test indicates that the CLL differences among the methods are significant at a level of 0.05 for each number of variables. This figure shows that the CLL gets smaller when more attributes are involved. With increasing the number of variables the random distribution becomes more *complex* (i.e., the number of dependencies among variables increases). However, we approximate the true distribution in any case with a 1-tree.

While we have shown empirically that our ordering heuristic tends to produce models that score the training data highly in the conditional likelihood sense, a higher conditional likelihood does not guarantee a higher accuracy, and training data results do not guarantee good generalization. In the

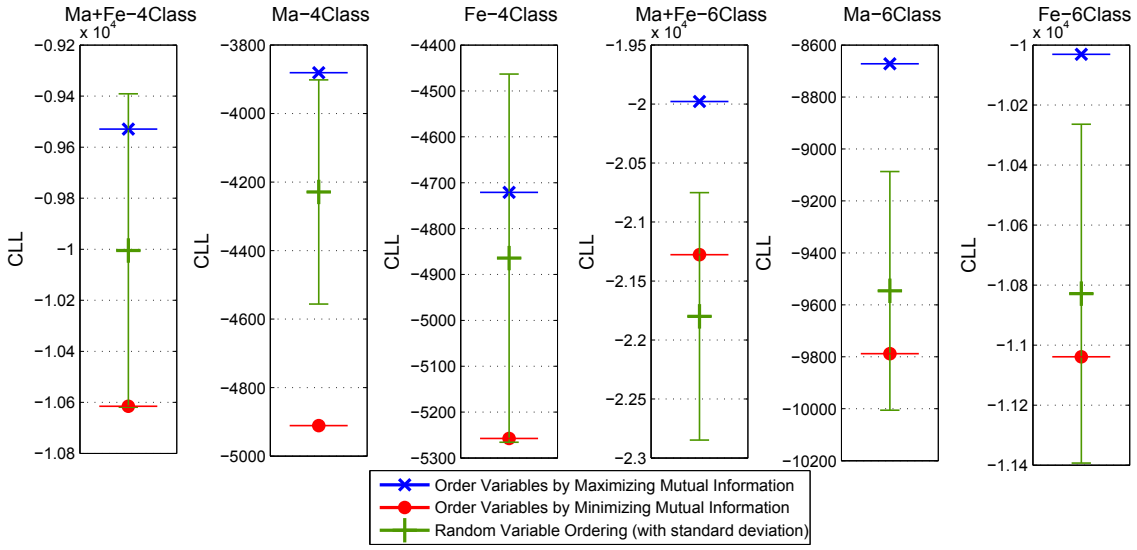


Figure 8: Resulting CLL on the TIMIT-4/6 data sets for a maximum mutual information (i.e., OMI), a minimum mutual information, and a random based ordering scheme.

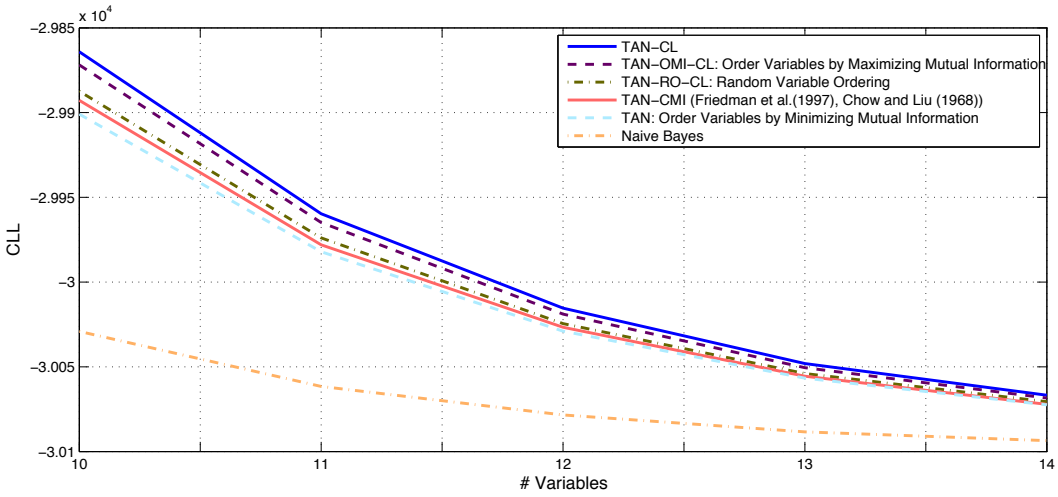


Figure 9: Optimized CLL of TAN structures learned by various algorithms. For each number of variables (x-axis) we generated 1000 random distributions.

next sections, however, we show that on balance, accuracy on test data using our ordering procedure is on par with the expensive greedy procedure, but with significantly less computation.

6.4 Synthetic Data

We show the benefit of the structure learning algorithms for the case where the class-dependent data are sampled from different 1-tree structures. In particular, we randomly determine for each class a 1-tree. The probabilities for each attribute variable are sampled from an uniform distribution, whereas the cardinality is set to 10, that is, $|X_i| = 10$. We use five classes. From the tree for $C = 1$ we draw 25000 samples. Additionally, we sample 6250 samples for each of the remaining four

classes from the same structure for confusion. For the remaining classes we draw 6250 samples from the corresponding random trees. This gives in total 75000 samples for training. The test set also consists of 75000 samples generated likewise. We perform this experiment for varying number of attributes, that is, $N \in \{5, 10, 15, 20, 25, 30\}$. The recognition results are shown in Figure 10, whereas the performance of each algorithm is averaged over 20 independent runs with randomly selected conditional probability distributions and trees. In each run, all algorithms have exactly the same data available.

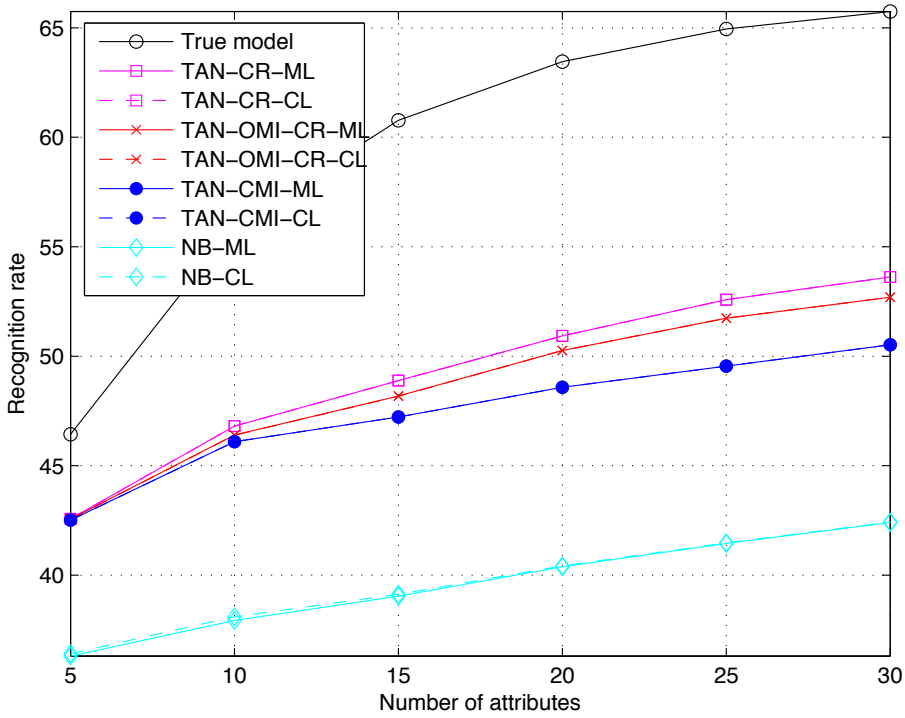


Figure 10: Synthetic data: Recognition performance is averaged over 20 runs.

We compare our OMI-CR heuristic to greedy discriminative structure learning. Additionally, we provide results for NB and generatively optimized TAN structures using CMI. To give a flavor about the data, the classification rates achieved with the true model used to generate the data are reported. This figure indicates that OMI-CR performs slightly worse than the greedy heuristic. However, the one-sided paired t-test (Mitchell, 1997) indicates that TAN-OMI-CR performs significantly better than TAN-CMI for more than 5 attributes at a level of 0.05. Generally, discriminative parameter optimization (i.e., CL) does not help for this data.

6.5 Classification Results and Discussion

Table 1 presents the averaged classification rates over the 25 UCI and 6 TIMIT-4/6 data sets.[4] Additionally, we report the CR on TIMIT-39, SurfInsp, MNIST, and USPS. The individual classification performance of all classification approaches on the 25 UCI data sets are summarized in Pernkopf

4. The average CR is determined by first weighting the CR of each data set with the number of samples in the test set. These values are accumulated and normalized by the total amount of samples in all test sets.

DATA SET / CLASSIFIER	UCI	TIMIT-4/6	TIMIT-39	SURFINSP	MNIST	USPS
NB-ML	81.50	84.85	61.70± 0.22	89.11 ± 1.47	83.73 ± 0.37	87.10 ± 0.61
NB-CL	85.18	**88.69**	**70.33** ± 0.20	92.67 ± 0.90	91.70 ± 0.28	93.67 ± 0.44
TAN-CMI-ML	84.82	86.18	65.40 ± 0.21	92.44 ± 0.96	91.28 ± 0.28	91.90 ± 0.50
TAN-CMI-CL	85.47	87.22	66.31 ± 0.21	92.44 ± 0.96	**93.80** ± 0.24	94.87 ± 0.40
TAN-RO-CR-ML (MEAN)	85.04	87.43	-	93.13 ± 0.70	-	-
TAN-RO-CR-ML (MIN)	85.00	87.57	-	92.67	-	-
TAN-RO-CR-ML (MAX)	84.82	87.43	-	92.67	-	-
TAN-SUPERPARENT-ML	84.80	87.54	66.53 ± 0.21	92.22 ±0.78	91.80 ± 0.27	90.67 ± 0.53
TAN-SUPERPARENT-CL	85.70	87.76	66.56 ± 0.21	92.44 ± 0.96	93.50 ± 0.25	94.70 ± 0.41
TAN-OMI-CR-ML	85.11	87.52	66.61± 0.21	**94.00** ±1.14	92.01 ± 0.27	92.40 ± 0.48
TAN-OMI-CR-CL	**85.82**	87.54	66.87 ± 0.21	**94.00** ±1.14	93.39 ± 0.25	94.90 ± 0.40
TAN-OMI-CRCL-ML	85.16	87.46	-	**94.22** ±1.13	-	-
TAN-OMI-CRCL-CL	85.78	87.62	-	**94.22** ±1.13	-	-
TAN-CR-ML	85.38	87.62	66.78 ± 0.21	92.89 ± 0.57	92.58 ± 0.26	92.57 ± 0.48
TAN-CR-CL	**86.00**	87.48	67.23 ± 0.21	92.89 ± 0.57	**93.94** ± 0.24	**95.83** ± 0.36
2-TREE-RO-CR-ML (MEAN)	-	87.86	-	-	-	-
2-TREE-RO-CR-ML (MIN)	-	87.87	-	-	-	-
2-TREE-RO-CR-ML (MAX)	-	87.87	-	-	-	
2-TREE-SUPERPARENT-ML	84.77	87.33	64.78 ± 0.21	92.67 ±1.63	90.56 ± 0.29	90.67 ± 0.53
2-TREE-SUPERPARENT-CL	**85.90**	87.14	67.38 ± 0.21	92.67 ±1.63	92.47 ± 0.26	94.13 ± 0.43
2-TREE-OMI-CR-ML	85.50	88.01	66.94 ± 0.21	**94.22** ± 0.82	92.69 ± 0.26	94.03 ± 0.41
2-TREE-OMI-CR-CL	**85.81**	87.27	67.06 ± 0.21	**94.88** ± 0.90	93.09 ± 0.25	94.76 ± 0.41
2-TREE-CR-ML	85.53	87.94	66.71 ± 0.21	**94.22** ± 1.07	-	-
2-TREE-CR-CL	85.73	86.95	67.36 ± 0.21	**94.22** ± 1.07	-	-

Table 1: Averaged classification results for 25 UCI and 6 TIMIT-4/6 data sets and classification results for TIMIT-39, SurfInsp, MNIST, and USPS with standard deviations. Best results use bold font. ML and CL denote generative and discriminative parameter learning, respectively. The order-based greedy heuristics are OMI-CR (order mutual information-CR) and RO-CR (random order-CR). CRCL refers to using discriminative parameter learning during structure learning. The generative structure learning algorithm is abbreviated as CMI and the greedy discriminative structure learning is TAN-CR and 2-tree-CR.

and Bilmes (2008a), whereas the random order experiment is presented in Table 8 (see Appendix A). For the TIMIT-4/6 data sets the individual classification performances for various classifier learning methods can be found in Table 9 (see Appendix B). The random order experiment for these data sets using a 2-tree is summarized in Appendix B in Table 10.

6.5.1 DISCUSSION: DISCRIMINATIVE VERSUS GENERATIVE PARAMETER LEARNING

Discriminative parameter learning produces mostly a significantly better classification performance than ML parameter learning on the same classifier structure. Especially, for cases where the structure of the underlying model is not optimized for classification (Greiner et al., 2005)—the average improvement of discriminative parameter learning over ML estimation on NB and generative TAN-CMI structures is large. In particular, for TIMIT-4/6 and TIMIT-39 the discriminatively optimized NB classifier (i.e., NB-CL) achieves the overall best classification performance. One reason is that the final step of MFCC features extraction includes a discrete cosine transform, that is, the features are decorrelated. Hence, the independence assumptions of the NB structure might be a good choice for these data sets. A second reason is that CL parameter learning for the TAN and 2-tree structures overfit the data (even with cross tuning)—the NB structure implicitly keeps the number of parameters low. Analyzing the results of CL parameter learning over various structures (especially for

2-trees) reveal that *cross tuning* is for some cases too restrictive concerning the number of conjugate gradient iterations—an alternative regularization method is required. CL parameter learning of NB classifiers is known to be equivalent to logistic regression. It can be shown that the CLL is concave when using $\log \theta^j_{i|h}$, that is, the global maximum can be found during discriminative parameter learning. Roos et al. (2005) showed that this also holds for more general network structures, for example, TAN.

In Section 6.6, we show that the classification performance of a discriminatively structured model may be superior to discriminatively parameterized models in the case of missing features.

6.5.2 DISCUSSION: DISCRIMINATIVE VERSUS GENERATIVE STRUCTURE LEARNING USING ML PARAMETER LEARNING

The CR objective function produces the best performing network structures. Evaluation of the CR measure is computationally expensive as mentioned above. However, due to the ordering of the variables in the order-based heuristics, we can reduce the number of CR evaluations from $O\left(N^3\right)$ to $O\left(N^2\right)$ for TAN structures. Hence, TAN-CR and 2-tree-CR are restricted to rather small data sets. The order-based heuristic OMI-CR achieve a similar performance at a much lower computational cost. Discriminative parameter learning during discriminative structure learning using our order-based heuristics can slightly improve the performance. This is possible only on small data sets due to the computational burden for the conjugate gradient parameter optimization.

The discriminative SuperParent algorithm performs slightly but not significantly worse compared to the other discriminative structure learning algorithms OMI-CR, OMI-CRCL, and greedy heuristic using CR on the UCI data set—similarly, the performance of SuperParent on TIMIT-4/6 for learning TAN structures, however, the performance using 2-tree structures is low. In summary, SuperParent achieves a lower classification rate compared to other discriminative structure learning algorithms on most of the data sets. The main reason for the degraded performance is an early termination of the algorithm.

For RO-CR we summarize the performance over 1000 random orderings using the mean (Mean), minimum (Min), and maximum (Max) CR (we use only 100 random orders for TIMIT-4/6 though). Min (Max) reports the classification rate on the test set using the structure which achieves the minimum (maximum) performance over 1000 random orderings (resp. 100 orders for TIMIT-4/6) on the training data. In some cases, the average over the data sets show that the worst RO-CR structures scored on the training sets perform better on the test sets than the best structures on the training sets, presumably due to overfitting. These results do show, however, that choosing from a collection of arbitrary orders and judging them based on the training set performance is not likely to perform well on the test set. Our heuristics do improve over these orders.

The TIMIT-39, MNIST, and USPS experiments show that we can perform discriminative structure learning for relatively large classification problems ($\sim$140000 samples, 66 features, 39 classes, $\sim$60000 samples, 196 features, 10 classes, and $\sim$8000 samples, 256 features, 10 classes, resp.). For these data sets, OMI-CR significantly outperform NB and TAN-CMI.

On MNIST we achieve a classification performance of $\sim$92.58% with the discriminative TAN classifier. A number of state-of-the-art algorithms, that is, convolutional net and virtual SVM, achieve an error rate below 1% (LeCun and Cortes). Due to resampling we use only 196 features in contrast to the 784 features of the original data set which might explain some of the loss in classification rate. Another reason why the convolutional neural net and virtual SVM perform better

CLASSIFIER STRUCTURE LEARNING PARAMETER LEARNING	TAN RO-CR ML	TAN SUPERPARENT ML	TAN OMI-CR ML	TAN OMI-CRCL ML	TAN CR ML	2-TREE SUPERPARENT ML	2-TREE OMI-CR ML	2-TREE CR ML
	MAX							
NB-ML	⇑0.0300	⇑0.0232	⇑0.0242	⇑0.0203	⇑0.0154	⇑0.0103	⇑0.0316	⇑0.0317
TAN-CMI-ML	↑0.120	⇐0.122	⇑0.0154	⇑0.0094	⇑0.0141	⇐0.0705	⇑0.0271	⇑0.0159
TAN-RO-CR-ML		⇐0.197	↑0.144	↑0.0945	⇑0.0446	⇐0.131	↑0.148	↑0.140
TAN-SUPERPARENT-ML			↑0.136	↑0.0848	↑0.0917	⇐0.189	↑0.149	↑0.139
TAN-OMI-CR-ML				↑0.182	↑0.190	⇐0.194	↑0.197	↑0.196
TAN-OMI-CRCL-ML					↑0.197	⇐0.182	↑0.196	↑0.197
TAN-CR-ML						⇐0.178	↑0.194	↑0.197
2-TREE-SUPERPARENT-ML							↑0.195	↑0.192
2-TREE-OMI-CR-ML								↑0.195

Table 2: Comparison of different classifiers using the one-sided paired t-test for the 25 UCI data sets: Each entry of the table gives the significance of the difference of the classification rate of two classifiers over the data sets. The arrow points to the superior learning algorithm. We use a double arrow if the difference is significant at the level of 0.05. The order-based greedy heuristics are OMI-CR (order mutual information-CR) and RO-CR (random order-CR). CRCL refers to algorithms using discriminative parameter learning during structure learning. The generative structure learning algorithm is abbreviated as CMI and the naive greedy discriminative structure learning is TAN-CR and 2-tree-CR.

CLASSIFIER STRUCTURE LEARN. PARAMETER LEARN.	2-TREE RO-CR ML	TAN SUPERPARENT ML	TAN OMI-CR ML	TAN OMI-CRCL ML	TAN CR ML	2-TREE SUPERPARENT ML	2-TREE OMI-CR ML	2-TREE CR ML
	MAX							
NB-ML	⇑<0.0001	⇑<0.0001	⇑<0.0001	⇑<0.0001	⇑<0.0001	⇑<0.0001	⇑<0.0001	⇑<0.0001
TAN-CMI-ML	⇑0.00032	⇑0.0012	⇑0.0016	⇑0.0019	⇑0.0012	⇑0.0027	⇑0.0002	⇑0.0002
2-TREE-RO-CR-ML		⇐0.0007	⇐0.0024	⇐0.0011	⇐0.0011	⇐0.0010	⇑0.0011	⇑0.0092
TAN-SUPERPARENT-ML			⇐0.189	⇐0.151	↑0.147	⇐0.0187	⇑0.0002	⇑0.0006
TAN-OMI-CR-ML				⇐0.078	↑0.140	⇐0.0078	⇑0.0004	⇑0.0013
TAN-OMI-CRCL-ML					⇑0.038	⇐0.054	⇑0.0002	⇑0.0007
TAN-CR-ML						⇐0.013	⇑0.0002	⇑0.0010
2-TREE-SUPERPARENT-ML							⇑0.0004	⇑0.0005
2-TREE-OMI-CR-ML								⇐0.069

Table 3: Comparison of different classifiers using the one-sided paired t-test for the 6 TIMIT-4/6 data sets: Each entry of the table gives the significance of the difference of the classification rate of two classifiers over the data sets. The arrow points to the superior learning algorithm. We use a double arrow if the difference is significant at the level of 0.05. The order-based greedy heuristics are OMI-CR (order mutual information-CR) and RO-CR (random order-CR). CRCL refers to algorithms using discriminative parameter learning during structure learning. The generative structure learning algorithm is abbreviated as CMI and the naive greedy discriminative structure learning is TAN-CR and 2-tree-CR.

on digit recognition is probably that images are not treated as unstructured feature vectors, that is, the convolutional neural net has built-in parts that look at particular areas of the image, and the virtual SVM is trained on augmented data that reflects invariance to small translations and rotations.

For the SurfInsp data the standard deviation of the five-fold cross-validation classification accuracy estimate is relatively large. Unfortunately, the size of the data set is limited to 450 samples.

The structure of Bayesian networks is implicitly regularized when we restrict the optimization over a model structure (e.g., 1-trees) assuming sufficient training data. For 2-trees we noticed that the data tended to overfit without further regularization. Therefore, we introduce 5-fold cross validation on the *training* data to find the optimal classifier structure.

Table 2 and Table 6.5.2 present a summary of the classification results over all structure learning experiments using ML parameter learning of the UCI and TIMIT-4/6 data sets.

We compare all pairs of classifiers using the one-sided paired t-test (Mitchell, 1997). The t-test determines whether the classifiers differ significantly under the assumption that the classification differences over the data set are independent and identically normally distributed. In these tables, each entry gives the significance of the difference in classification rate of two classification approaches. The arrow points to the superior learning algorithm and a double arrow indicates whether the difference is significant at a level of 0.05.

These tables show that TAN-OMI-CR, TAN-OMI-CRCL-CR, and TAN-CR significantly outperform the generative structure learning approach TAN-CMI (similar for 2-trees). However, the computationally expensive greedy heuristic TAN-CR and 2-tree-CR do not significantly outperform our discriminative order-based heuristics TAN-OMI-CR and 2-tree-OMI-CR, respectively.

6.5.3 DISCUSSION: COMPUTATIONAL REQUIREMENTS FOR STRUCTURE LEARNING

The running time of the TAN-CMI, TAN-OMI-CR, and TAN-CR structure learning algorithms for the UCI, TIMIT-4/6, TIMIT-39, and the MNIST data sets is summarized in Table 4. The numbers represent the percentage of time that is needed for a particular algorithm compared to TAN-CR. TAN-CMI is roughly 3-10 times faster than TAN-OMI-CR and TAN-CR takes about 10-40 times longer for establishing the discriminative structure than TAN-OMI-CR.

Data set	TAN-CMI	TAN-OMI-CR	TAN-CR
UCI	0.649%	3.155%	100.00%
TIMIT-4/6	3.56%	11.47%	100.00%
TIMIT-39	0.11%	2.08%	100.00%
MNIST	0.21%	2.23%	100.00%

Table 4: Running time of structure learning algorithms relative to TAN-CR.

6.6 Results with Randomly Missing Input Features

As mentioned in Section 3, generative models can easily deal with missing features simply by marginalizing out from the model the missing feature. We are particularly interested in a testing context which has arbitrary sets of missing features for each classification sample. In such a case, it is not possible to re-train the model for each potential set of missing features without also memorizing the training set. Due to the local-normalization property of Bayesian networks and the structure of any model with a parent-less class node, marginalization is as easy as an $O(r^{k+1})$ operation for a k-tree, where r is the domain size of each feature.

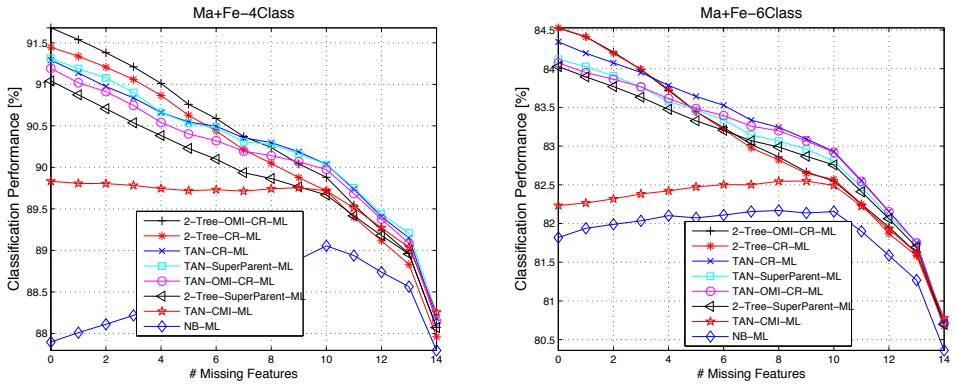


Figure 11: Classification performance of different structure learning methods assuming missing features using Ma+Fe data of TIMIT-4/6.

In Figure 11, we present the classification performance of discriminative and generative structures assuming missing features using the Ma+Fe data of TIMIT-4/6. The x-axis denotes the number of missing features. The curves are averaged over 100 classifications of the test data with uniformly at random selected missing features. We use exactly the same missing features for each classifier. Variance bars are omitted to improve readability, but indicate that the resulting differences are significant between NB-ML, TAN-CMI-ML, and discriminatively structured classifiers for a low number of missing features. Hence, discriminatively structured Bayesian network classifiers outperform TAN-CMI-ML even in the case of missing features. This demonstrates, at least empirically, that discriminatively structured generative models do not lose their ability to impute missing features.

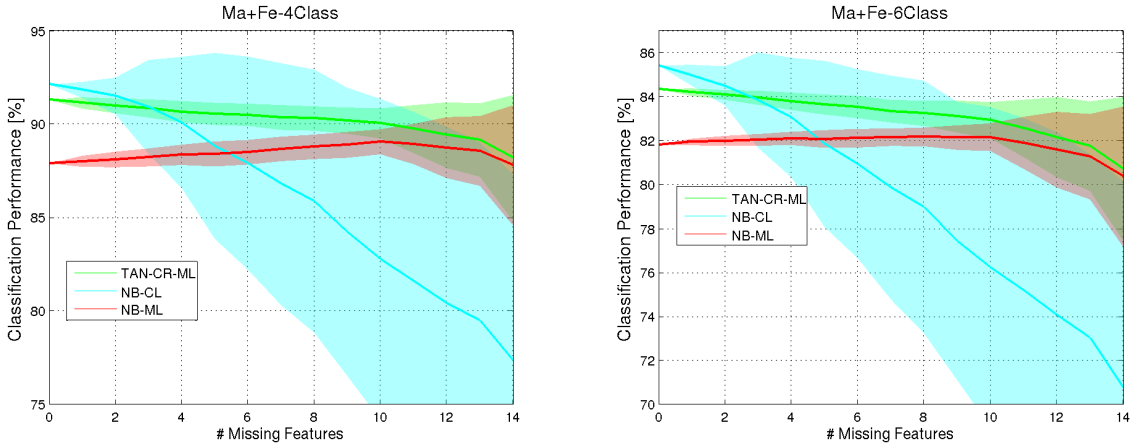


Figure 12: Classification performance of NB-ML, NB-CL, and TAN-CR-ML assuming missing features using Ma+Fe data of TIMIT-4/6. The shaded region corresponds to the standard deviation over 100 classifications.

In Figure 12, we show for the same data sets and experimental setup that the classification performance of a discriminatively structured model may be superior to discriminatively parameterized models in the case of missing features. In particular, for more than three missing features TAN-CR-ML outperforms NB-CL. Similar results can be shown for MNIST in Figure 13.

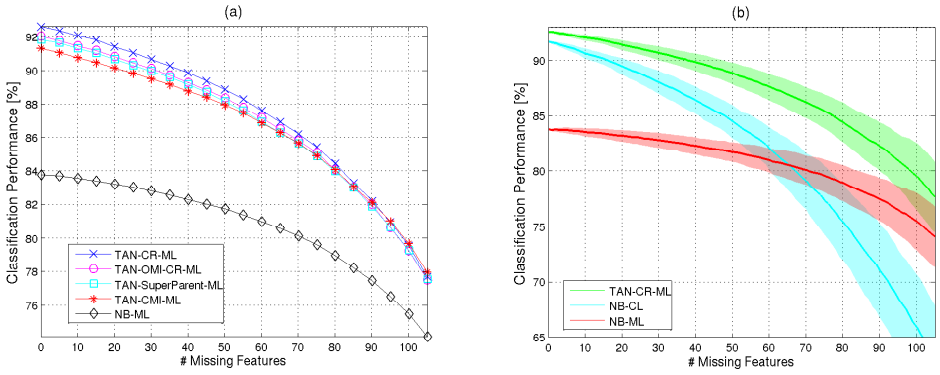


Figure 13: Classification performance using MNIST. The shaded region corresponds to the standard deviation over 100 classifications: (a) Different structure learning methods, (b) NB-ML, NB-CL, and TAN-CR-ML.

6.7 Results with SVMs

In Table 6, we compare classification performances between the best performing Bayesian network classifiers (in Table 1) and SVMs using RBF kernels. SVMs outperform our discriminative Bayesian network classifier. For TIMIT-4/6 one reason might be that SVMs are applied to the continuous feature domain. In Table 5 we compare the model complexity (i.e., number of parameters) between both SVMs and Bayesian network classifiers. This table reveals that the Bayesian network uses $\sim$108, $\sim$66, $\sim$212, and $\sim$259 times fewer parameters for MNIST, USPS, Ma+Fe-4, and Ma+Fe-6 than the SVM. This might also explain the loss in classification performance of discriminative Bayesian networks. Furthermore, Bayesian network classifiers can be directly applied to problems with more than two classes, whereas SVMs in its traditional formulation are limited to binary problems—the multiclass problem is decomposed into binary problems. Additionally, for SVMs we have to select C^* and σ. A substantial difference is that SVMs determine the number of support vectors automatically while in the case of Bayesian networks the number of parameters is specified by the structure. A limited complexity class (e.g., 1-tree) restricts the number of parameters which might be advantageous. In contrast to SVMs, a Bayesian network might be preferred since it is easy to work with missing features (see Section 6.6), parameter tying and knowledge-based hierarchical decomposition is facilitated, and it is easy to work with structured data.

DATA SET	TIMIT-4/6	MNIST	USPS
CLASSIFIER			
NB-CL	88.69		
TAN-CR-CL		93.94 ± 0.24	95.83 ± 0.36
SVM	**89.38**	**96.40** ± 0.19	**97.86** ± 0.26
PARAMETERS	$C^* = 1, \sigma = 0.05$	$C^* = 1, \sigma = 0.01$	$C^* = 1, \sigma = 0.005$

Table 5: Model complexity for best Bayesian network (BN) and SVM.

7. Conclusion

We introduced a simple order-based heuristic for learning discriminative network structure. The metric for establishing the ordering of N features is based on either the conditional mutual infor-

DATA SET	N	NUMBER OF SVS	NUMBER OF SVM PARAMETERS	NUMBER OF BN PARAMETERS
MNIST	196	17201	3371396	31149
USPS	256	3837	982272	14689
TIMIT-4/6 (MA+FE-4)	20	13146	262920	1239
TIMIT-4/6 (MA+FE-6)	20	24350	487000	1877

Table 6: Classification results for TIMIT-4/6 data sets (averaged), MNIST, and USPS using best performing Bayesian network classifier (see Table 1) and SVMs.

mation or the classification rate. Given an ordering, we can find a discriminative classifier structure using $O\left(N^{k+1}\right)$ score evaluations (where constant k is the tree-width of the sub-graph over the attributes). We empirically compare the performance of our algorithms to state-of-the-art discriminative and generative parameter and structure learning algorithms using real data from the TIMIT speech corpus, the UCI repository, a visual surface inspection task, and from handwritten digit recognition tasks. The experiments show that the discriminative structures found by our order-based heuristics achieve on average a significantly better classification performance than the generative approach. Our obtained classification performance is very similar to the greedy search using CR. Our order-based heuristics however, are about 10-40 times faster. Additionally, we show that discriminatively structured Bayesian network classifiers are superior even in the case of missing features.

Acknowledgments

We would like to acknowledge support for this project from the Austrian Science Fund (Project number P22488-N23) and (Project number S10604-N13).

Appendix A. UCI data

	DATA SET	# FEATURES	# CLASSES	# SAMPLES TRAIN	# SAMPLES TEST
1	AUSTRALIAN	14	2	690	CV-5
2	BREAST	10	2	683	CV-5
3	CHESS	36	2	2130	1066
4	CLEVE	13	2	296	CV-5
5	CORRAL	6	2	128	CV-5
6	CRX	15	2	653	CV-5
7	DIABETES	8	2	768	CV-5
8	FLARE	10	2	1066	CV-5
9	GERMAN	20	2	1000	CV-5
10	GLASS	9	7	214	CV-5
11	GLASS2	9	2	163	CV-5
12	HEART	13	2	270	CV-5
13	HEPATITIS	19	2	80	CV-5
14	IRIS	4	3	150	CV-5
15	LETTER	16	26	15000	5000
16	LYMPHOGRAPHY	18	4	148	CV-5
17	MOFN-3-7-10	10	2	300	1024
18	PIMA	8	2	768	CV-5
19	SHUTTLE-SMALL	9	7	3866	1934
20	VOTE	16	2	435	CV-5
21	SATIMAGE	36	6	4435	2000
22	SEGMENT	19	7	1540	770
23	SOYBEAN-LARGE	35	19	562	CV-5
24	VEHICLE	18	4	846	CV-5
25	WAVEFORM-21	21	3	300	4700

Table 7: 25 UCI data sets

CLASSIFIER	OMI-CR	RO-CR			
DATA SET		MEAN ± STD	MEDIAN	MIN	MAX
AUSTRALIAN	82.04	84.58 ± 0.77	84.62	84.63	83.60
BREAST	97.40	97.19 ± 0.26	97.23	97.39	96.95
CHESS	94.93	94.70 ± 0.76	94.75	95.22	93.90
CLEVE	81.76	81.42 ± 0.97	81.42	82.76	82.76
CORRAL	99.20	95.79 ± 2.55	96.17	96.80	95.20
CRX	84.07	84.21 ± 0.82	84.23	84.07	84.22
DIABETES	74.36	75.23 ± 0.52	75.26	75.53	75.01
FLARE	82.74	82.38 ± 0.50	82.44	81.80	81.98
GERMAN	73.20	72.83 ± 0.87	72.80	71.50	71.50
GLASS	70.70	71.74 ± 1.13	71.81	70.20	73.44
GLASS2	82.20	81.94 ± 0.73	82.14	82.14	82.20
HEART	81.85	82.83 ± 0.85	82.96	82.59	83.33
HEPATITIS	90.67	89.25 ± 1.69	89.00	90.33	90.33
IRIS	93.33	93.58 ± 0.55	93.33	92.67	93.33
LETTER	87.00	86.53 ± 0.60	86.54	86.70	86.50
LYMPHOGRAPHY	88.30	85.83 ± 1.72	85.92	87.12	81.27
MOFN-3-7-10	91.41	90.11 ± 1.01	90.14	89.55	89.75
PIMA	75.26	75.70 ± 0.49	75.65	75.78	76.56
SHUTTLE-SMALL	99.17	99.33 ± 0.13	99.33	99.22	99.17
VOTE	94.29	93.90 ± 0.69	93.84	93.36	94.06
SATIMAGE	88.25	87.47 ± 0.44	87.45	87.05	87.20
SEGMENT	94.42	94.63 ± 0.59	94.68	92.99	95.45
SOYBEAN-LARGE	91.11	92.29 ± 0.78	92.34	92.14	91.74
VEHICLE	67.38	68.18 ± 0.90	68.13	67.42	68.20
WAVEFORM-21	78.02	78.19 ± 0.51	78.21	78.77	77.79
AVERAGE	85.11	85.04	85.06	85.00	84.82

Table 8: Classification results in $[\%]$ with RO-CR compared to OMI-CR for the UCI data. Min (Max) reports the CR on the test set using the structure which achieves the minimum (maximum) performance over 1000 random orderings on the training data.

Appendix B. TIMIT-4/6 Data

DATA SET NUMBER OF CLASSES CLASSIFIER	MA+FE 4	MA 4	FE 4	MA+FE 6	MA 6	FE 6	AVERAGE
NB-ML	87.90 ± 0.18	88.69 ± 0.25	87.67 ± 0.25	81.82 ± 0.20	82.26 ± 0.28	81.93 ± 0.28	84.85
NB-CL	**92.12** ± 0.15	**92.81** ± 0.20	**91.57** ± 0.22	**85.41** ± 0.18	**86.28** ± 0.26	**85.12** ± 0.26	**88.69**
TAN-CMI-ML	89.83 ± 0.17	90.20 ± 0.23	90.36 ± 0.23	82.23 ± 0.20	83.20 ± 0.28	82.99 ± 0.28	86.18
TAN-CMI-CL	90.96 ± 0.16	91.39 ± 0.22	90.92 ± 0.22	83.06 ± 0.20	84.85 ± 0.27	84.05 ± 0.27	87.22
TAN-SUPERPARENT-ML	91.31 ± 0.15	91.84 ± 0.21	90.71 ± 0.23	84.12 ± 0.19	84.84 ± 0.27	83.51 ± 0.27	87.54
TAN-SUPERPARENT-CL	91.56 ± 0.15	92.29 ± 0.21	90.74 ± 0.22	84.36 ± 0.19	84.84 ± 0.27	83.83 ± 0.27	87.76
TAN-OMI-CR-ML	91.19 ± 0.16	92.15 ± 0.21	90.51 ± 0.23	84.07 ± 0.19	84.68 ± 0.27	83.71 ± 0.27	87.52
TAN-OMI-CR-CL	91.37 ± 0.15	92.28 ± 0.21	90.51 ± 0.23	84.00 ± 0.19	84.49 ± 0.27	83.75 ± 0.27	87.54
TAN-OMI-CRCL-ML	91.09 ± 0.16	91.99 ± 0.21	90.35 ± 0.23	84.05 ± 0.19	84.59 ± 0.27	83.87 ± 0.27	87.46
TAN-OMI-CRCL-CL	91.41 ± 0.15	92.55 ± 0.21	90.54 ± 0.23	83.88 ± 0.19	84.71 ± 0.27	84.08 ± 0.27	87.62
TAN-CR-ML	91.29 ± 0.16	91.81 ± 0.21	90.52 ± 0.23	84.35 ± 0.19	84.80 ± 0.27	83.93 ± 0.27	87.62
TAN-CR-CL	91.29 ± 0.16	92.04 ± 0.21	90.52 ± 0.23	83.69 ± 0.19	84.83 ± 0.27	83.91 ± 0.27	87.48
2-TREE-SUPERPARENT-ML	91.02 ± 0.16	91.84 ± 0.21	90.52 ± 0.23	84.01 ± 0.19	84.22 ± 0.27	83.42 ± 0.27	87.33
2-TREE-SUPERPARENT-CL	90.39 ± 0.16	91.51 ± 0.22	90.62 ± 0.23	83.17 ± 0.20	85.30 ± 0.26	83.96 ± 0.27	87.14
2-TREE-OMI-CR-ML	91.68 ± 0.15	92.28 ± 0.21	91.03 ± 0.22	84.52 ± 0.19	85.43 ± 0.26	84.31 ± 0.27	88.01
2-TREE-OMI-CR-CL	91.28 ± 0.16	91.79 ± 0.21	90.53 ± 0.23	83.46 ± 0.19	84.48 ± 0.27	83.42 ± 0.27	87.27
2-TREE-CR-ML	91.45 ± 0.15	92.22 ± 0.21	91.11 ± 0.22	84.53 ± 0.19	85.36 ± 0.26	84.22 ± 0.27	87.94
2-TREE-CR-CL	91.00 ± 0.16	91.41 ± 0.22	90.52 ± 0.23	82.79 ± 0.20	84.46 ± 0.27	83.19 ± 0.28	86.95

Table 9: Classification results in [%] for 4 and 6 classes with standard deviation. Best results use bold font. ML and CL denote generative and discriminative parameter learning, respectively. OMI-CR (order mutual information-CR) refers to the order-based greedy heuristic. OMI-CRCL refers to OMI-CR using discriminative parameter learning during structure learning. The generative structure learning algorithm is abbreviated as CMI and the greedy discriminative structure learning is TAN-CR and 2-tree-CR.

CLASSIFIER		2-TREE-OMI-CR	2-TREE-RO-CR			
DATA SET	NUMBER OF CLASSES		MEAN $\pm$ STD	MEDIAN	MIN	MAX
MA+FE	4	91.68	91.50 ± 0.10	91.49	91.44	91.46
MA	4	92.28	92.23 ± 0.10	92.23	92.35	92.14
FE	4	91.03	90.95 ± 0.13	90.94	90.77	90.94
MA+FE	6	84.52	84.44 ± 0.13	84.45	84.64	84.49
MA	6	85.43	85.17 ± 0.17	85.18	85.06	85.30
FE	6	84.31	84.04 ± 0.16	84.05	83.96	84.11
AVERAGE		**88.01**	87.86	87.86	87.87	87.87

Table 10: Classification results in [%] with 2-tree-RO-CR compared to 2-tree-OMI-CR for 4 and 6 classes. Min (Max) reports the CR on the test set using the structure which achieves the minimum (maximum) performance over 100 random orderings on the training data.

References

S. Acid, L.M. de Campos, and J.G. Castellano. Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. *Machine Learning*, 59:213–235, 2005.

S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Methods*, 8(2):277–284, 1987.

L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer. Maximum Mutual Information estimation of HMM parameters for speech recognition. In *IEEE Intern. Conf. on Acoustics, Speech, and Signal Processing*, pages 49–52, 1986.

P.L. Bartlett, M.I. Jordan, and J.D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

J. Bilmes. *Natural Statistical Models for Automatic Speech Recognition*. PhD thesis, U.C. Berkeley, 1999.

J. Bilmes. Dynamic Bayesian multinets. In *16th Inter. Conf. of Uncertainty in Artificial Intelligence (UAI)*, pages 38–45, 2000.

J. Bilmes, G. Zweig, T. Richardson, K. Filali, K. Livescu, P. Xu, K. Jackson, Y. Brandman, E. Sandness, E. Holtz, J. Torres, and B. Byrne. Discriminatively structured graphical models for speech recognition: JHU-WS-2001. Technical report, Johns Hopkins University, 2001.

C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

C.M. Bishop and J. Lasserre. Generative or discriminative? Getting the best of both worlds. *Bayesian Statistics*, 8:3–23, 2007.

N. Brenner, S. Strong, R. Koberle, and W. Bialek. Synergy in a neural code. *Neural Computation*, 12:1531–1552, 2000.

W.L. Buntine. Theory refinement on Bayesian networks. In *7th Conference on Uncertainty in AI (UAI)*, pages 52–60, 1991.

C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transaction on Information Theory*, 14:462–467, 1968.

G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

R.G Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer Verlag, 1999.

S. Dasgupta. The sample complexity of learning fixed-structure Bayesian networks. *Machine Learning*, 29(2):165–180, 1997.

L.M. de Campos. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7:2149–2187, 2006.

P. Domingos and M.J. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *12th International Conference on Machine Learning (ICML)*, pages 194–202, 1995.

Y. Ephraim and L.R. Rabiner. On the releations between modeling approaches for speech recognition. *IEEE Transactions on Information Theory*, 36(2):372–380, 1990.

Y. Ephraim, A. Dembo, and L.R. Rabiner. A minimum discrimination information approach for Hidden Markov Models. *IEEE Transactions on Information Theory*, 35(5):1001–1013, 1989.

U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29: 131–163, 1997.

N. Friedman, I. Nachman, and D. Peer. Learning Bayesian network structure form massive datasets: The sparse candidate algorithm. In *15th Conference on Uncertainty in AI (UAI)*, pages 196–205, 1999.

D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.

R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. In *18th Conf. of the AAAI*, pages 167–173, 2002.

R. Greiner, X. Su, S. Shen, and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59:297–322, 2005.

A. Gretton and L. Gÿorfi. Nonparametric independence tests: Space partitioning and kernel approaches. In *Algorithmic Learning Theory: 19th International Conference (ALT08)*, pages 183–198, 2008.

D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *21st Inter. Conf. of Machine Lerning (ICML)*, pages 361–368, 2004.

A.K. Halberstadt and J. Glass. Heterogeneous measurements for phonetic classification. In *Proceedings of EUROSPEECH*, pages 401–404, 1997.

D. Heckerman. A tutorial on learning Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

G. Heigold, T. Deselaers, R. Schlüter, and H. Ney. Modified MMI/MPE: A direct evaluation of the margin in speech recognition. In *Intern. Conf. on Machine learning (ICML)*, pages 384–391, 2008.

T. Jebara. *Discriminative, Generative and Imitative Learning*. PhD thesis, Media Laboratory, MIT, 2001.

M.I. Jordan. *Learning in Graphical Models*. MIT Press, 1999.

B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, 40(12):3043–3054, 1992.

B.-H. Juang, W. Chou, and C.-H. Lee. Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 5(3):257–265, 1997.

D.R. Karger and N. Srebro. Learning Markov networks: Maximum bounded tree-width graphs. In *Symposium on Discrete Algorithms*, pages 302–401, 2001.

E.J. Keogh and M.J. Pazzani. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *7th International Workshop on Artificial Intelligence and Statistics*, pages 225–230, 1999.

R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

J.B. Kruskal. On the shortest spanning subtree and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, volume 7, pages 48–50, 1956.

L. Lamel, R. Kassel, and S. Seneff. Speech database development: Design and analysis of the acoustic-phonetic corpus. In *Proceedings of the DARPA Speech Recognition Workshop, Report No. SAIC-86/1546*, 1986.

S.L. Lauritzen. *Graphical Models*. Oxford Science Publications, 1996.

Y. LeCun and C. Cortes. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

C. Meek. Causal inference and causal explanation with background knowledge. In *11th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 403–410, 1995.

C. Merz, P. Murphy, and D. Aha. UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, 1997. `www.ics.uci.edu/\~{}mlearn/MLRepository.html`.

T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

K.P. Murphy. *Dynamic Bayesian networks: Representation, Iference and Learning*. PhD Thesis, University of California, Berkeley, 2002.

N. Narasimhan and J. Bilmes. A supermodular-submodular procedure with applications to discriminative structure learning. In *21st Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2005.

A.Y. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems 14*, 2002.

M. Pazzani. Searching for dependencies in Bayesian classifiers. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 239–248, 1996.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

F. Pernkopf. Detection of surface defects on raw steel blocks using Bayesian network classifiers. *Pattern Analysis and Applications*, 7(3):333–342, 2004.

F. Pernkopf. Bayesian network classifiers versus selective k-NN classifier. *Pattern Recognition*, 38 (3):1–10, 2005.

F. Pernkopf and J. Bilmes. Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *Intern. Conf. on Machine Learning (ICML)*, pages 657 – 664, 2005.

F. Pernkopf and J. Bilmes. Efficient heuristics for discriminative structure learning of Bayesian network classifiers. Technical report, Laboratory of Signal Processing and Speech Communication, Graz University of Technology, 2008a.

F. Pernkopf and J.A. Bilmes. Order-based discriminative structure learning for Bayesian network classifiers. In *International Symposium on Artificial Intelligence and Mathematics*, 2008b.

F. Pernkopf and M. Wohlmayr. On discriminative parameter learning of bayesian network classifiers. In *European Conference on Machine Learning (ECML)*, pages 221–237, 2009.

F. Pernkopf, T. Van Pham, and J.A. Bilmes. Broad phonetic classification using discriminative Bayesian networks. *Speech Communication*, 143(1):123–138, 2008.

W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C*. Cambridge Univ. Press, 1992.

R. Raina, Y. Shen, A.Y Ng, and A. McCallum. Classification with hybrid generative/discriminative models. In *Advances in Neural Information Processing Systems 16*, 2004.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59:267–296, 2005.

B. Schölkopf and A.J. Smola. *Learning with kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

N. Srebro. Maximum likelihood bounded tree-width Markov networks. *Artificial Intelligence*, 143 (1):123–138, 2003.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *21th Conference on Uncertainty in AI (UAI)*, pages 584 – 590, 2005.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1): 267–288, 1996.

V. Vapnik. *Statistical Learning Theory*. Wiley & Sons, 1998.

X. Zhang, L. Song, A. Gretton, and A. Smola. Kernel measures of independence for non-iid data. In *Advances on Neural Information Processing Systems 22*, 2009.

Journal of Machine Learning Research 11 (2010) 2361-2386 Submitted 8/08; Revised 8/09; Published 8/10

High-dimensional Variable Selection with Sparse Random Projections: Measurement Sparsity and Statistical Efficiency

Dapo Omidiran DAPO@EECS.BERKELEY.EDU
Martin J. Wainwright* WAINWRIG@EECS.BERKELEY.EDU
Department of Electrical Engineering and Computer Sciences
UC Berkeley
Berkeley, CA 94720

Editor: Sanjoy Dasgupta

Abstract

We consider the problem of high-dimensional variable selection: given n noisy observations of a k-sparse vector $\beta^* \in \mathbb{R}^p$, estimate the subset of non-zero entries of β^*. A significant body of work has studied behavior of ℓ_1-relaxations when applied to random measurement matrices that are dense (e.g., Gaussian, Bernoulli). In this paper, we analyze *sparsified* measurement ensembles, and consider the trade-off between measurement sparsity, as measured by the fraction γ of non-zero entries, and the statistical efficiency, as measured by the minimal number of observations n required for correct variable selection with probability converging to one. Our main result is to prove that it is possible to let the fraction on non-zero entries $\gamma \to 0$ at some rate, yielding measurement matrices with a vanishing fraction of non-zeros per row, while retaining the same statistical efficiency as dense ensembles. A variety of simulation results confirm the sharpness of our theoretical predictions.

Keywords: variable selection, sparse random projections, high-dimensional statistics, Lasso, consistency, ℓ_1-regularization

1. Introduction

Recent years have witnessed a flurry of research on the recovery of high-dimensional models satisfying some type of sparsity constraint. These types of sparse recovery problems arise in a variety of domains, including variable selection in regression (Tibshirani, 1996), graphical model selection (Meinshausen and Buhlmann, 2006; Ravikumar et al., 2010), sparse principal components analysis (Johnstone and Lu, 2009; Paul, 2007), sparse approximation (Tropp, 2006), and compressed sensing (Candes and Tao, 2005; Donoho, 2006). In all of these settings, the basic problem is to recover information about a high-dimensional signal $\beta^* \in \mathbb{R}^p$, based on a set of n observations. The signal β^* is assumed *a priori* to be sparse: either exactly k-sparse, or lying within some ℓ_q-ball with $q < 1$.

A particular instance of high-dimensional sparse recovery involves the linear regression model $Y = X\beta^* + W$, where $Y \in \mathbb{R}^n$ is the observation vector, $W \in \mathbb{R}^n$ is observation noise, and $X \in \mathbb{R}^{n \times p}$ is the measurement matrix. In this context, high-dimensional scaling means that the sample size n can be of the same order of magnitude, or substantially smaller than the ambient dimension p. A particularly simple version of this model, studied extensively within the compressed sensing com-

*. Also in the Department of Statistics.

munity (Candes and Tao, 2005; Donoho, 2006), is the *noiseless version* in which $W = 0$, so that the model reduces to the under-determined linear system $Y = X\beta^*$. Here the model is only interesting when $n \ll p$, so that the linear system is not fully determined. Other work in machine learning and statistics (e.g., Meinshausen and Buhlmann, 2006; Zhao and Yu, 2006; Wainwright, 2009; Zhou et al., 2007) has focused on the noisy version of the model, say with $W \sim N(0, \sigma^2 I_p)$ for some noise variance σ^2. The problem of high-dimensional regression can be studied either for deterministic designs, for which the measurement matrix X is fixed, or for random designs in which X is drawn randomly from some ensemble. Past work on random designs has focused on the behavior of various ℓ_1-relaxations when applied to measurement matrices drawn from the standard Gaussian ensemble (e.g., Donoho, 2006; Candes and Tao, 2005), or more general random ensembles satisfying mutual incoherence conditions (Meinshausen and Buhlmann, 2006; Wainwright, 2009). A measurement matrix X drawn from such a standard random ensemble is dense, in that each row of X has p non-zero entries with high probability. In various applications of the sparse regression problem, the measurement matrix is itself a design variable, and dense measurement matrices are undesirable. For instance, in applications such as sensor networks (Wang et al., 2007), digital imaging (Wakin et al., 2006) or database management (Achlioptas, 2001; Li et al., 2006), it would be preferable to take measurements of the signal β^* based on sparse inner products, using measurement matrices X in which each row has a relatively small fraction of non-zero entries. Furthermore, sparse measurement matrices require significantly less storage space, and have the potential for reduced algorithmic complexity for signal recovery, since many algorithms for linear programming and conic programming more generally (Boyd and Vandenberghe, 2004), can be accelerated by exploiting problem structure.

In the noiseless instance of the regression problem (with $n \ll p$), the standard approach to estimating β^* is by solving the basis pursuit linear program (Chen et al., 1998). Recent work by Baraniuk et al. (2007) has established connections between the success of this method and the behavior of random projections, as characterized by the Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss, 1984; Dasgupta and Gupta, 2003). Random projections and their applications have been studied extensively in machine learning and related fields, with applications to dimensionality reduction (Dasgupta, 1999; Li et al., 2007), data stream processing (Alon et al., 1996; Indyk, 2006), databases (Achlioptas, 2001; Li et al., 2006) and compressed sensing (Wang et al., 2007; Baraniuk et al., 2007). One standard proof of the Johnson-Lindenstrauss lemma is based on random Gaussian matrices. Achlioptas (2001) was the first to apply sparse random projections, with each entry distributed on $\{-1, 0, 1\}$ with probabilities $[\frac{1}{6}, \frac{2}{3}, \frac{1}{6}]$, to the Johnson-Lindenstrauss problem setting, and to provide the same guarantees as dense projections. Unfortunately, his proof technique does not allow the non-zero mass to be decreased much beyond $\frac{1}{3}$. Other recent work (Li et al., 2006) provides theoretical and experimental justification for scaling the non-zero mass aggressively to zero. Stemming from different sources than the random projection literature, another line of recent work (e.g., Cormode and Muthukrishnan, 2005; Gilbert et al., 2006; Sarvotham et al., 2006; Xu and Hassibi, 2007) has studied compressed sensing methods based on sparse measurement matrices, using constructions motivated by group testing and coding theory.

Whereas this past work focuses on the noiseless recovery problem, our primary interest in this paper is the noisy linear observation model which, as we show, exhibits qualitatively different behavior than the noiseless case. At a high level, our primary goal in this paper is not to design sparse measurement matrices, but rather to gain a theoretical understanding of the *trade-off between the*

degree of measurement sparsity, and statistical efficiency. We assess measurement sparsity in terms of the fraction γ of non-zero entries in any particular row of the measurement matrix, and we define statistical efficiency in terms of the minimal number of measurements n required to recover the correct support with probability converging to one. Our interest can be viewed in terms of experimental design: more precisely we ask, what degree of measurement sparsity can be permitted without increasing the number of observations required for correct variable selection or subset recovery?

To bring sharp focus to the issue, we analyze this question for exact subset recovery using ℓ_1-constrained quadratic programming, also known as the Lasso in the statistics literature (Chen et al., 1998; Tibshirani, 1996), where past work on dense Gaussian measurement ensembles (Wainwright, 2009) provides a precise characterization of its success/failure. We characterize the density of our measurement ensembles with a positive parameter $\gamma \in (0,1]$, corresponding to the fraction of non-zero entries per row. We first show that for all fixed $\gamma \in (0,1]$, the statistical efficiency of the Lasso remains the same as with dense measurement matrices. We then prove that it is possible to let $\gamma \to 0$ at some rate, as a function of the sample size n, signal length p and signal sparsity k, yielding measurement matrices with a vanishing fraction of non-zeroes per row while requiring exactly the same number of observations as dense measurement ensembles. In general, in contrast to the noiseless setting (Xu and Hassibi, 2007), our theory still requires that the average number of non-zeroes per column of the measurement matrix (i.e., γn) tend to infinity, however, under the loss function considered here (exact signed support recovery), we prove that no method can succeed with probability one if this condition does not hold.

The remainder of this paper is organized as follows. In Section 2, we set up the problem more precisely, state our main result, and discuss some of its implications. In Section 3, we provide a high-level outline of the proof with more technical aspects of the argument deferred to the appendices. We provide some illustrative simulations in Section 4 that illustrate the sharpness of our theoretical predictions. Work in this paper was presented in part at the International Symposium on Information Theory in Toronto, Canada (July, 2008). We note that concurrent and complementary work (Wang et al., 2010) analyzes the information-theoretic limitations of sparse measurement matrices for exact support recovery.

1.1 Notation

Throughout this paper, we use the following standard asymptotic notation: $f(n) = O(g(n))$ if $f(n) \leq Cg(n)$ for some constant $C < +\infty$; $f(n) = \Omega(g(n))$ if $f(n) \geq cg(n)$ for some constant $c > 0$; and $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. In addition, we use $\log(x)$ to denote the natural logarithm of x.

2. Problem Set-up and Main Result

We begin by setting up the problem, stating our main result, and discussing some of its consequences.

2.1 Observation Model

Define the *support set* of a signal $\beta \in \mathbb{R}^p$

$$S(\beta) \quad := \quad \{i \in \{1,\ldots,p\} \mid \beta_i \neq 0\},$$

and consider the class of k-sparse signals of length p:

$$\mathcal{C}(p,k,\beta^*_{\min}) \quad := \quad \left\{\beta \in \mathbb{R}^p \mid |S(\beta)| = k \leq \frac{p}{2}, \min_{i \in S} |\beta_i| \geq \beta^*_{\min}\right\}. \tag{1}$$

Let $\beta^* \in \mathbb{R}^p$ be a fixed but unknown vector in $\mathcal{C}(p,k,\beta^*_{\min})$, and suppose that we make a set $\{Y_1,\ldots,Y_n\}$ of n independent and identically distributed (i.i.d.) observations of the unknown vector β^*, each of the form

$$Y_i \quad := \quad x_i^T \beta^* + W_i, \tag{2}$$

where $W_i \sim \mathcal{N}(0,1)$ is observation noise, and $x_i \in \mathbb{R}^p$ is a measurement vector. Note that there is no loss in generality in assuming that the noise variance is one, since the observation model with signal class $\mathcal{C}(p,k,\beta^*_{\min})$ and $W_i \sim \mathcal{N}(0,1)$ is equivalent to the observation model with signal class $\mathcal{C}(p,k,\frac{\beta^*_{\min}}{\sigma})$ with noise variance $W_i \sim \mathcal{N}(0,\sigma^2)$.

It is convenient to use $Y = \begin{bmatrix} Y_1 & Y_2 & \ldots & Y_n \end{bmatrix}^T$ to denote the n-vector of measurements, with similar notation for the noise vector $W \in \mathbb{R}^n$, and

$$X \quad = \quad \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} \quad = \quad \begin{bmatrix} X_1 & X_2 & \ldots & X_p \end{bmatrix}.$$

to denote the $n \times p$ measurement matrix. With this notation, the observation model can be written compactly as $Y = X\beta^* + W$.

2.2 Sign Consistency and Statistical Efficiency

Given some estimate $\widehat{\beta}$, its error relative to the true β^* can be assessed in various ways, depending on the underlying application of interest. For applications in compressed sensing, various types of ℓ_r norms (i.e., $\mathbb{E}\|\widehat{\beta} - \beta^*\|_r^r$) are well-motivated, whereas for statistical prediction, it is most natural to study a predictive loss (e.g., $\|X(\widehat{\beta} - \beta^*)\|_2^2/n$). For reasons of scientific interpretation or for model selection purposes, the object of primary interest is the support S of β^*. In this paper, we consider a slightly stronger notion of model selection: in particular, our goal is to recover the *signed support* of the unknown β^*, as defined by the p-vector $S_+(\beta^*)$ with elements

$$[S_+(\beta^*)]_i \quad := \quad \begin{cases} \operatorname{sign}(\beta^*_i) & \text{if } \beta^*_i \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Given some estimate $\widehat{\beta}$, we study the probability $\mathbb{P}[S_+(\widehat{\beta}) = S_+(\beta^*)]$ that it correctly specifies the signed support. In particular, a sequence of estimates $\widehat{\beta}_{n,p,k}$ is *sign consistent* if

$$\mathbb{P}[S_+(\widehat{\beta}_{n,p,k}) = S_+(\beta^*)] \to 1, \text{ as } n,p,k \to \infty. \tag{3}$$

The estimator that we analyze is ℓ_1-constrained quadratic programming (QP), also known as the Lasso (Tibshirani, 1996) in the statistics literature. The Lasso generates an estimate $\widehat{\beta}$ by solving the regularized QP

$$\widehat{\beta} \quad = \quad \arg\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2n}\|Y - X\beta\|_2^2 + \rho_n \|\beta\|_1 \right\}, \tag{4}$$

where $\rho_n > 0$ is a user-defined regularization parameter. A large body of past work has focused on the model selection behavior of the Lasso for both deterministic and random measurement matrices (e.g., Tropp, 2006; Zhao and Yu, 2006; Wainwright, 2009).

Tropp (2006) demonstrates that under some technical conditions the Lasso produces an estimate with support contained in the true support set, while Zhao and Yu (2006) prove the sign consistency of the Lasso for certain sequences (n, k_n, p_n) and measurement ensemble covariances. The main contribution of Wainwright (2009) is to identify the smallest n required for sign consistency of the Lasso when applied to measurement matrices X drawn randomly from Gaussian ensembles, with the standard Gaussian ensemble (i.e., each element $X_{ij} \sim \mathcal{N}(0,1)$ i.i.d.) being one special case. Define the function $n_{crit}(p,k) := 2k\log(p-k)$. The paper (Wainwright, 2009) shows the Lasso undergoes a phase transition as a function of the control parameter

$$\theta(n,p,k) \quad := \quad \frac{n}{n_{crit}(p,k)}. \tag{5}$$

In more detail, for the special case of the standard Gaussian ensemble, for any sequence (n,p,k) such that $\theta(n,p,k) > 1+\varepsilon$ for some $\varepsilon > 0$, the Lasso (with an appropriate choice of regularization parameter ρ_n) is sign consistent with probability converging to one. In contrast, it fails to be sign consistent with high probability, regardless of the choice of ρ_n, for any sequences such that $\theta(n,p,k) < 1-\varepsilon$.

The main contribution of this paper is to show that sparse measurement ensembles are *statistically efficient*: the same sharp threshold (5) holds for γ-sparsified measurement ensembles, including a subset for which $\gamma \to 0$, so that each row of the measurement matrix has a vanishing fraction of non-zero entries.

2.3 Statement of Main Result

A measurement matrix $X \in \mathbb{R}^{n\times p}$ drawn randomly from a Gaussian ensemble is dense, in that each row has $\Theta(p)$ non-zero entries. The main focus of this paper is the observation model (2), using measurement ensembles that are designed to be sparse. To formalize the notion of sparsity, we let $\gamma \in (0,1]$ represent a *measurement sparsity parameter*, corresponding to the (average) fraction of non-zero entries per row. Our analysis allows the sparsity parameter $\gamma(n,p,k)$ to be a function of the triple (n,p,k), but we typically suppress this explicit dependence so as to simplify notation. For a given choice of γ, we consider measurement matrices X with i.i.d. entries of the form

$$X_{ij} \quad \stackrel{d}{=} \quad \begin{cases} Z \sim \mathcal{N}(0,1) & \text{with probability } \gamma \\ 0 & \text{with probability } 1-\gamma. \end{cases} \tag{6}$$

By construction, the expected number of non-zero entries in each row of X is γp. In fact, the analysis of this paper establishes exactly the same control parameter threshold (5) for γ-sparsified measurement ensembles, for any fixed $\gamma \in (0,1)$, as the completely dense case ($\gamma = 1$). In particular, we state the following result on conditions under which the Lasso applied to sparsified ensembles has the same *sample complexity* as when applied to the dense (standard Gaussian) ensemble:

Theorem 1 *Suppose that the measurement matrix $X \in \mathbb{R}^{n\times p}$ is drawn with i.i.d. entries according to the γ-sparsified distribution* (6)*. Then for any $\varepsilon > 0$, if the sample size satisfies*

$$n \quad > \quad (2+\varepsilon)k\log(p-k), \tag{7}$$

then the Lasso is sign consistent as $(n,p,k) \to +\infty$ so long as

$$\frac{n\rho_n^2\gamma}{\log(p-k)} \quad \to \quad \infty, \tag{8}$$

$$\frac{\rho_n}{\beta^*_{\min}} \max\Big\{1, \frac{\sqrt{k}}{\gamma}\sqrt{\frac{\log\log(p-k)}{\log(p-k)}}\Big\} \quad \to \quad 0, \tag{9}$$

$$\gamma^3 \min\Big\{k, \frac{\log(p-k)}{\log\log(p-k)}\Big\} \quad \to \quad \infty. \tag{10}$$

Remark 2 *(a) Note that the sample complexity* (7) *is identical to the Lasso threshold proven in past work (Wainwright, 2009). To gain intuition on the remaining conditions, it is helpful to consider various special cases of the sparsity parameter γ. If γ is a constant fixed to some value in $(0,1]$, then it plays no role in the scaling, and condition* (10) *is always satisfied. Conditions* (8) *and* (9) *are slightly weaker than the corresponding condition from previous work, in that they require that ρ_n be slightly larger, and hence that $\beta^*_{\min}$ must approach zero more slowly than the requirement of this previous work. Depending on the exact behavior of $\beta^*_{\min}$, choosing ρ_n^2 to decay slightly more slowly than $\log p/n$ is sufficient to guarantee exact recovery with $n = \Theta(k\log(p-k))$, meaning that we recover exactly the same statistical efficiency as the dense case ($\gamma = 1$) for all constant measurement sparsities $\gamma \in (0,1)$. At least initially, one might think that reducing γ should increase the required number of observations, since it effectively reduces the signal-to-noise ratio by a factor of γ. However, under high-dimensional scaling ($p \to +\infty$), a major effect limiting the Lasso performance is the number $(p-k)$ of irrelevant factors, and under the scaling considered here, this effect is dominant.*

(b) However, Theorem 1 also allows for general scalings of the measurement sparsity γ along with the triplet (n,p,k). More concretely, let us suppose for simplicity that $\beta^*_{\min} = \Theta(1)$. Then over a range of signal sparsities—say $k = \alpha p$, $k = \Theta(\sqrt{p})$ or $k = \Theta(\log(p-k))$, corresponding respectively to linear sparsity, polynomial sparsity, and exponential sparsity—we can choose a decaying measurement sparsity, for instance

$$\gamma \quad = \quad \Big[\frac{\log\log(p-k)}{\log(p-k)}\Big]^{\frac{1}{6}} \to 0 \tag{11}$$

along with the regularization parameter $\rho_n^2 = \frac{\log(p-k)}{n}\sqrt{\frac{\log(p-k)}{\log\log(p-k)}}$ while maintaining the same sample complexity (required number of observations for support recovery) as the Lasso with dense measurement matrices.

(c) Of course, the conditions of Theorem 1 do not allow the measurement sparsity γ to approach zero arbitrarily quickly. Rather, for any γ guaranteeing exact recovery, condition (8) *implies that the average number of non-zero entries per column of X (namely, γn) must tend to infinity. (Indeed, with $n = \Omega(k\log(p-k))$, our specific choice* (11) *certainly satisfies this constraint.) A natural question is whether exact recovery is possible using measurement matrices, either randomly drawn or deterministically designed, with the average number of non-zeros per column (namely γn) remaining bounded. In fact, under the criterion of exactly recovering the signed support* (3)*, as shown by the following result, if $\beta^*_{\min} = O(1)$, then no method can succeed with probability converging to one unless γn tends to infinity.*

Proposition 1 *If $\gamma n[\beta^*_{\min}]^2$ does not tend to infinity, then no method can recover the signed support with probability one.*

Proof We construct a sub-problem that must be solvable by any method capable of performing exact signed support recovery. Suppose that $\beta^*_1 = \beta^*_{\min} \neq 0$ and that the column X_1 has n_1 non-zero entries, say without loss of generality indices $i = 1, \ldots, n_1$. Now consider the problem of recovering the sign of β^*_1. Let us extract the observations $i = 1, \ldots, n_1$ that explicitly involve β^*_1, writing

$$Y_i \quad = \quad X_{i1}\beta^*_1 + \sum_{j \in T(i)} X_{ij}\beta^*_j + W_i, \qquad i = 1, \ldots, n_1 \tag{12}$$

where $T(i)$ denotes the set of indices in row i for which X_{ij} is non-zero, excluding index 1. Even assuming that $\{\beta^*_j, j \in T(i)\}$ were perfectly known, this observation model (12) is at best equivalent to observing β^*_1 contaminated by constant variance additive Gaussian noise, and our task is to distinguish whether $\beta^*_1 = \beta^*_{\min}$ or $\beta^*_1 = -\beta^*_{\min}$. The average $\bar{Y} = \frac{1}{n_1}\sum_{i=1}^{n_1}[Y_i - \sum_{j \in T(i)} X_{ij}\beta^*_j]$ is a sufficient statistic, following the distribution $\bar{Y} \sim \mathcal{N}(\beta^*_{\min}, \frac{1}{n_1})$. Unless the effective signal-to-noise ratio, which is of the order $n_1[\beta^*_{\min}]^2$, goes to infinity, there will always be a constant probability of error in distinguishing $\beta^*_1 = \beta^*_{\min}$ from $\beta^*_1 = -\beta^*_{\min}$. Under the γ-sparsified random ensemble, we have $n_1 \leq (1 + o(1))\gamma n$ with high probability, so that no method can succeed unless $\gamma n[\beta^*_{\min}]^2$ goes to infinity, as claimed. ■

Note that the conditions in Theorem 1 imply that $n\gamma[\beta^*_{\min}]^2 \to +\infty$. In particular, condition (9) implies that $\rho^2_n = o([\beta^*_{\min}]^2)$, and condition (8) implies that $n\gamma\rho^2_n \to +\infty$, which verifies the condition of Proposition 1.

3. Proof of Theorem 1

This section is devoted to the proof of Theorem 1. We begin with a high-level outline of the proof; as with previous work on dense Gaussian ensembles (Wainwright, 2009), the key is the notion of a *primal-dual witness* for exact signed support recovery. The proof itself involves a number of additional steps not needed in this past work, in order to gain good control on sparse matrices as opposed to generic Gaussian matrices (see Appendix D). The proof is divided into a sequence of separate lemmas, with some of the more technical results deferred to the appendices.

3.1 High-level Overview of Proof

For the purposes of our proof, it is convenient to consider matrices $X \in \mathbb{R}^{n \times p}$ with i.i.d. entries of the form

$$X_{ij} \quad \stackrel{d}{=} \quad \begin{cases} Z \sim \mathcal{N}(0, \frac{1}{\gamma}) & \text{with probability } \gamma \\ 0 & \text{with probability } 1 - \gamma. \end{cases}$$

So as to obtain an equivalent observation model, we also reset the variance of each noise term W_i to be $\frac{1}{\gamma}$. Finally, we can assume without loss of generality that $S_+(\beta^*_S) = \vec{1} \in \mathbb{R}^k$.

Define the *sample covariance matrix*

$$\widehat{\Sigma} \quad := \quad \frac{1}{n}X^T X \; = \; \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T.$$

Of particular importance to our analysis is the $k \times k$ sub-matrix $\widehat{\Sigma}_{SS}$. For future reference, we state the following claim, proved in Appendix D:

Lemma 1 *Under the conditions of Theorem 1, the submatrix $\widehat{\Sigma}_{SS}$ is invertible with probability greater than $1 - O(\frac{1}{(p-k)^2})$.*

The foundation of our proof is the following lemma: it provides sufficient conditions for the Lasso (4) to recover the signed support set.

Lemma 2 (Primal-dual conditions for support recovery) *Suppose that $\widehat{\Sigma}_{SS} \succ 0$, and that we can find a primal vector $\widehat{\beta} \in \mathbb{R}^p$, and a subgradient vector $\widehat{z} \in \mathbb{R}^p$ that satisfy the* zero-subgradient condition

$$\widehat{\Sigma}\big(\widehat{\beta} - \beta^*\big) - \frac{1}{n} X^T W + \rho_n \widehat{z} \;\; = \;\; 0, \tag{13}$$

and the signed-support-recovery conditions

$$\widehat{z}_i \;\; = \;\; \mathrm{sign}(\beta^*_i) \qquad \textit{for all } i \in S, \tag{14}$$

$$\widehat{\beta}_j \;\; = \;\; 0 \qquad \textit{for all } j \in S^c, \tag{15}$$

$$|\widehat{z}_j| \;\; < \;\; 1 \qquad \textit{for all } j \in S^c, \textit{ and} \tag{16}$$

$$\mathrm{sign}(\widehat{\beta}_i) \;\; = \;\; \mathrm{sign}(\beta^*_i) \qquad \textit{for all } i \in S. \tag{17}$$

Then $\widehat{\beta}$ is the unique optimal solution to the Lasso (4)*, and recovers the correct signed support.*

See Appendix B.1 for the proof of this claim.

On the basis of Lemmas 1 and 2, it suffices to show that under the specified scaling of (n, p, k), there exists a primal-dual pair $(\widehat{\beta}, \widehat{z})$ satisfying the conditions of Lemma 2. We establish the existence of such a pair with the following constructive procedure:

(a) We begin by setting $\widehat{\beta}_{S^c} = 0$, and $\widehat{z}_S = \mathrm{sign}(\beta^*_S)$.

(b) Next we determine $\widehat{\beta}_S$ by solving the linear system

$$\widehat{\Sigma}_{SS}\big(\widehat{\beta}_S - \beta^*_S\big) - \frac{1}{n} X_S^T W + \rho_n \,\mathrm{sign}(\beta^*_S) \;\; = \;\; 0.$$

(c) Finally, we determine $\widehat{z}_{S^c}$ by solving the linear system:

$$-\rho_n \widehat{z}_{S^c} \;\; = \;\; \widehat{\Sigma}_{S^c S}\big(\widehat{\beta}_S - \beta^*_S\big) - \frac{1}{n} X_{S^c}^T W.$$

By construction, this procedure satisfies the zero sub-gradient condition (13), as well as auxiliary conditions (14) and (15); it remains to verify conditions (16) and (17).

In order to complete these final two steps, it is helpful to define for $i \in S$ and $j \in S^c$ the following random variables:

$$V_j^a \;\; := \;\; X_j{}^T \big[\frac{1}{n} X_S (\widehat{\Sigma}_{SS})^{-1} \vec{1}\big] \rho_n, \tag{18}$$

$$V_j^b \;\; := \;\; X_j{}^T \big[I_{n\times n} - \frac{1}{n} X_S (\widehat{\Sigma}_{SS})^{-1} X_S^T\big] \frac{W}{n}, \quad \text{and} \tag{19}$$

$$U_i \;\; := \;\; e_i^T \widehat{\Sigma}_{SS}^{-1} \big[\frac{1}{n} X_S^T W - \rho_n \vec{1}\big], \tag{20}$$

where $e_i \in \mathbb{R}^k$ is the unit vector with one in position i, and $\vec{1} \in \mathbb{R}^k$ is the all-ones vector.

A little bit of algebra (see Appendix B.2 for details) shows that $\rho_n \widehat{z}_j = V_j^a + V_j^b$, and that $U_i = \widehat{\beta}_i - \beta_i^*$. Consequently, if we define the events

$$\mathcal{E}(V) \quad := \quad \big\{ \max_{j \in S^c} |V_j^a + V_j^b| < \rho_n \big\}, \tag{21}$$

$$\mathcal{E}(U) \quad := \quad \big\{ \max_{i \in S} |U_i| \leq \beta^*_{\min} \big\}, \tag{22}$$

where $\beta^*_{\min}$ was defined previously as the minimum value of $|\beta^*|$ on its support, then in order to establish that the Lasso succeeds in recovering the exact signed support, it suffices to show that $\mathbb{P}[\mathcal{E}(V) \cap \mathcal{E}(U)] \to 1$,

We decompose the proof of this final claim in the following three lemmas. As in the statement of Theorem 1, suppose that $n > (2+\varepsilon) k \log(p-k)$, for some fixed $\varepsilon > 0$.

Lemma 3 (Control of V^a) *Under the conditions of Theorem 1, there exists a fixed positive value δ (dependent on ε) such that*

$$\mathbb{P}[\max_{j \in S^c} |V_j^a| \geq (1-\delta)\rho_n] \quad \to \quad 0.$$

Lemma 4 (Control of V^b) *Under the conditions of Theorem 1, there exists a fixed positive value δ (dependent on ε)*

$$\mathbb{P}[\max_{j \in S^c} |V_j^b| \geq \delta \rho_n] \quad \to \quad 0.$$

Lemma 5 (Control of U) *Under the conditions of Theorem 1, we have*

$$\mathbb{P}[(\mathcal{E}(U))^c] \; = \; \mathbb{P}[\max_{i \in S} |U_i| > \beta^*_{\min}] \quad \to \quad 0.$$

3.2 Proof of Lemma 3

We assume throughout that $\widehat{\Sigma}_{SS}$ is invertible, an event which occurs with probability $1 - o(1)$ under the stated assumptions (see Lemma 1). If we define the n-dimensional vector

$$h \quad := \quad X_S (\widehat{\Sigma}_{SS})^{-1} \vec{1}, \tag{23}$$

then the variable V_j^a can be written compactly as

$$\frac{V_j^a}{\rho_n} \; = \; X_j^T h = \sum_{\ell=1}^n h_\ell X_{\ell j}.$$

Note that each term $X_{\ell j}$ in this sum is distributed as a mixture variable, taking the value 0 with probability $1-\gamma$, and distributed as $\mathcal{N}(0, \frac{1}{\gamma})$ variable with probability γ. For $\ell = 1, \ldots, n$ and each j, define the random vector H^j, with entries

$$H_\ell^j \quad \stackrel{d}{=} \quad \begin{cases} h_\ell & \text{with probability } \gamma \\ 0 & \text{with probability } 1-\gamma. \end{cases}$$

For each index $\ell = 1, \ldots, n$, let $Z_{\ell j} \sim \mathcal{N}(0, \frac{1}{\gamma})$. With these definitions, by construction, we have that

$$\frac{V_j^a}{\rho_n} \stackrel{d}{=} \sum_{\ell=1}^n H_\ell^j Z_{\ell j}.$$

To gain some intuition for the behavior of this sum, note that the variables $\{Z_{\ell j}, \ell = 1, \ldots, n\}$ are independent of the vector H^j. (In particular, H^j is a function of X_S, whereas $Z_{\ell j}$ is a function of $X_{\ell j}$, with $j \notin S$.) Consequently, we may condition on H^j without affecting Z, and since Z is Gaussian, we have $(\frac{V_j^a}{\rho_n} \mid H^j) \sim \mathcal{N}(0, \frac{\|H^j\|_2^2}{\gamma})$. Therefore, if we can obtain good control on the norm $\|H^j\|_2$, then we can use standard Gaussian tail bounds (see Appendix A) to control the maximum $\max_{j \in S^c} V_j^a / \rho_n$. The following lemma is proved in Appendix C:

Lemma 6 *Under condition* (10)*, then for any fixed* $\delta > 0$*, we have*

$$\mathbb{P}\big[\|H^j\|_2^2 \leq \frac{\gamma k(1+\delta)}{n}\big] \geq 1 - O\big[\exp(-\min\{2\log(p-k), \frac{n}{2k}\})\big].$$

The primary implication of the above bound is that each V_j^a / ρ_n variable is (essentially) no larger than a $\mathcal{N}(0, \frac{k}{n})$ variable. We can then use standard techniques for bounding the tails of Gaussian variables to obtain good control over the random variable $\max_{j \in S^c} |V_j^a| / \rho_n$. In particular, by the union bound, we have

$$\mathbb{P}[\max_{j \in S^c} |V_j^a| \geq (1-\delta)\rho_n] \leq (p-k)\, \mathbb{P}[\sum_{\ell=1}^n H_\ell^j Z_{\ell j} \geq (1-\delta)].$$

For any $\delta > 0$, define the event $\mathcal{T}^j(\delta) := \{\|H^j\|_2^2 \leq \frac{k\gamma(1+\delta)}{n}\}$. With this definition, we have

$$\begin{aligned}
\mathbb{P}[\max_{j \in S^c} |V_j^a| \geq (1-\delta)\rho_n] &\leq (p-k)\left\{\mathbb{P}[\sum_{\ell=1}^n H_\ell^j Z_{\ell j} \geq (1-\delta) \mid \mathcal{T}^j(\delta)] + \mathbb{P}[(\mathcal{T}^j(\delta)^c)]\right\} \\
&\leq (p-k)\left\{2\exp(-\frac{n(1-\delta)^2}{2k(1+\delta)}) + c_1 \exp\big(-\min\big(2\log(p-k), \frac{n}{2k}\big)\big)\right\},
\end{aligned}$$

where the second inequality uses a standard Gaussian tail bound (see Appendix A), and Lemma 6. Finally, let us assume the condition $n > (2+\varepsilon)k\log(p-k)$ for some fixed $\varepsilon > 0$. Then there exists a numerical constant c_1 such that

$$\begin{aligned}
\mathbb{P}[\max_{j \in S^c} |V_j^a| \geq (1-\delta)\rho_n] &\leq (p-k)\left\{2\exp\big(-\frac{n(1-\delta)^2}{2k(1+\delta)}\big) + c_1 \exp(-\min\big(2\log(p-k), \frac{n}{2k}\big))\right\} \\
&= (p-k)\left\{2\exp\big(-\frac{n(1-\delta)^2}{2k(1+\delta)}\big) + c_1 \exp(-2\log(p-k))\right\} \\
&\leq (p-k)\left\{2\exp\big(-(2+\varepsilon)\log(p-k)\frac{(1-\delta)^2}{2(1+\delta)}\big)\right. \\
&+ \left. c_1 \exp(-2\log(p-k))\right\}.
\end{aligned}$$

Note that the above inequality holds for all values of ε. Since $\varepsilon > 0$ is fixed, we can choose a fixed value of δ such that $\frac{(1-\delta)^2}{1+\delta} > \frac{2}{2+\varepsilon/2}$. With this choice, we then have

$$\begin{aligned}
\mathbb{P}[\max_{j \in S^c} |V_j^a| \geq (1-\delta)\rho_n] &\leq (p-k)\left\{2\exp\big(-\frac{(2+\varepsilon)}{(2+\varepsilon/2)}\log(p-k)\big) + c_1 \exp(-2\log(p-k))\right\} \\
&\leq (p-k)\left\{(2+c_1)\exp\big(-\frac{(2+\varepsilon)}{(2+\varepsilon/2)}\log(p-k)\big)\right\},
\end{aligned}$$

thereby establishing that $\mathbb{P}[\max_{j\in S^c}|V_j^a| \geq (1-\delta)\rho_n] \to 0$, as claimed.

3.3 Proof of Lemma 4

Defining the orthogonal projection matrix $\Pi_S^{\perp} := I_{n\times n} - X_S(X_S^T X_S)^{-1}X_S^T$, we then have

$$\begin{aligned}\mathbb{P}[\max_{j\in S^c}|V_j^b| \geq \delta\rho_n] &= \mathbb{P}[\max_{j\in S^c}\big|X_j^T\Pi_S^{\perp}(\frac{W}{n})\big| \geq \delta\rho_n] \\ &\leq (p-k)\,\mathbb{P}\big[\big|X_j^T\Pi_S^{\perp}(\frac{W}{n})\big| \geq \delta\rho_n\big]. \end{aligned} \tag{24}$$

Now since $\Pi_S^{\perp}$ is an orthogonal projection matrix, and the column vector X_j, noise vector W and randomness in $\Pi_S^{\perp}$ are all independent, we have the bound

$$\mathbb{P}\big[\big|X_j^T\Pi_S^{\perp}(W/n)\big| \geq \delta\rho_n\big] \leq \mathbb{P}\big[\big|X_j^T(W/n)\big| \geq \delta\rho_n\big], \tag{25}$$

For each $\ell = 1,\ldots,n$ and $j = 1,\ldots,p$, let $B_{\ell j}$ be a Bernoulli variable with parameter γ, and let $Z_{\ell j} \sim \mathcal{N}(0,\frac{1}{\gamma})$. In terms of these random variables, we have the representation $X_{\ell j} = B_{\ell j}Z_{\ell j}$. Note moreover that the the sum $\sum_{\ell=1}^n B_{\ell j}$ is a binomial random variable, and define the event

$$\mathcal{T} := \big\{\frac{1}{n}\big|\sum_{\ell=1}^n B_{\ell j} - \gamma n\big| \leq \frac{1}{2\sqrt{k}}\big\}.$$

From the Hoeffding bound (see Lemma 7), we have $\mathbb{P}[\mathcal{T}^c] \leq 2\exp(-\frac{n}{2k})$. Using the representation $X_{\ell j} = B_{\ell j}Z_{\ell j}$ and conditioning on $\mathcal{T}$, we obtain

$$\begin{aligned}\mathbb{P}\big[\big|X_j^T W/n\big| \geq \delta\rho_n\big] &\leq \mathbb{P}\big[\big|\frac{1}{n}\sum_{\ell=1}^n B_{\ell j}Z_{\ell j}W_\ell\big| \geq \delta\rho_n \mid \mathcal{T}\big] + \mathbb{P}[\mathcal{T}^c] \\ &\leq \mathbb{P}\big[\big|\frac{1}{n}\sum_{\ell=1}^{n(\gamma+\frac{1}{2\sqrt{k}})} Z_{\ell j}W_\ell\big| \geq \delta\rho_n\big] + 2\exp(-\frac{n}{2k}),\end{aligned}$$

where we have assumed without loss of generality that the first $n(\gamma+\frac{1}{2\sqrt{k}})$ variables in the collection $\{B_{\ell j}\}_{\ell=1}^n$ are non-zero.

Conditioned on W, the random variable $M_j := \frac{1}{n}\sum_{\ell=1}^{n(\gamma+\frac{1}{2\sqrt{k}})} Z_{\ell j}W_\ell$ is zero-mean Gaussian with variance $\nu(W;\gamma) := \frac{1}{n^2\gamma}\sum_{\ell=1}^{n(\gamma+\frac{1}{2\sqrt{k}})} W_\ell^2$. For some $\delta_1 > 0$, define the event

$$\mathcal{T}_2(\delta_1) := \Big\{\nu(W;\gamma) \leq (1+\delta_1)\frac{1}{n\gamma^2}\,(\gamma+\frac{1}{2\sqrt{k}})\Big\}.$$

In order to bound the probability of this event, we begin by observing that since $W_\ell \sim N(0,1/\gamma)$, the variable $n^2\gamma^2\nu(W;\gamma)$ is chi-squared with $d = n(\gamma+\frac{1}{2\sqrt{k}})$ degrees of freedom. Consequently, using χ^2-tail bounds (see Appendix A), we have

$$\mathbb{P}[(\mathcal{T}_2(\delta_1))^c] \leq \exp\big(-n(\gamma+\frac{1}{2\sqrt{k}})\frac{3\delta_1^2}{16}\big).$$

Now, by conditioning on $\mathcal{T}_2(\delta_1)$ and its complement and using tail bounds on Gaussian variates (see Appendix A), we obtain

$$\begin{aligned}\mathbb{P}\big[\big|\frac{1}{n}\sum_{\ell=1}^{n(\gamma+\frac{1}{2\sqrt{k}})} Z_{\ell j}W_\ell\big| \geq \delta\rho_n\big] &\leq \mathbb{P}\big[\big|\frac{1}{n}\sum_{\ell=1}^{n(\gamma+\frac{1}{2\sqrt{k}})} Z_{\ell j}W_\ell\big| \geq \delta\rho_n \mid \mathcal{T}_2(\delta_1)\big] + \mathbb{P}[(\mathcal{T}_2(\delta_1))^c] \\ &\leq 2\exp\big(-\frac{n\gamma^2(\delta^2\rho_n^2)}{2(1+\delta_1)(\gamma+\frac{1}{2\sqrt{k}})}\big) + \\ &\qquad \exp\big(-n(\gamma+\frac{1}{2\sqrt{k}})\frac{3\delta_1^2}{16}\big). \end{aligned} \tag{26}$$

Finally, putting together the pieces from Equations (26), (25), and (24) we obtain that $\mathbb{P}[\max_{j\in S^c}|V_j^b| \geq \delta\rho_n]$ is upper bounded by

$$(p-k)\left\{2\exp(-\frac{n}{2k}) + 2\exp\big(-\frac{n\gamma^2(\delta^2\rho_n^2)}{2(1+\delta_1)(\gamma+\frac{1}{2\sqrt{k}})}\big) + \exp\big(-n(\gamma+\frac{1}{2\sqrt{k}})\frac{3\delta_1^2}{16}\big)\right\}. \tag{27}$$

The first term in Equation (27) goes to zero since $n > (2+\varepsilon)k\log(p-k)$. Condition (10) implies that $\gamma\sqrt{k} \to \infty$. In particular, this means that eventually $1 \leq 2\gamma\sqrt{k}$. Once this occurs, we have the inequality $\frac{\gamma^2}{\gamma+\frac{1}{2\sqrt{k}}} > \frac{\gamma}{2}$, and hence

$$\begin{aligned}(p-k)\exp\big(-\frac{n\gamma^2(\delta^2\rho_n^2)}{2(1+\delta_1)(\gamma+\frac{1}{2\sqrt{k}})}\big) &\leq (p-k)\exp\big(-\frac{n\gamma\delta^2\rho_n^2}{4(1+\delta_1)}\big) \\ &= (p-k)\exp\big(-\log(p-k)\frac{n\gamma\delta^2\rho_n^2}{4(1+\delta_1)\log(p-k)}\big).\end{aligned}$$

Recalling that the terms δ and δ_1 are fixed constants, condition (10) implies that eventually

$$(p-k)\exp\big(-\log(p-k)\frac{n\gamma\delta^2\rho_n^2}{4(1+\delta_1)\log(p-k)}\big) \leq (p-k)\exp\big(-3\log(p-k)\big),$$

showing that the middle term of Equation (27) goes to zero. Finally, using the condition $n \geq (2+\varepsilon)k\log(p-k)$, we obtain

$$\begin{aligned}(p-k)\exp\big(-n(\gamma+\frac{1}{2\sqrt{k}})\frac{3\delta_1^2}{16}\big) &\leq (p-k)\exp\big(-n(\frac{1}{2\sqrt{k}})\frac{3\delta_1^2}{16}\big) \\ &\leq (p-k)\exp\big(-\big[\frac{(2+\varepsilon)\sqrt{k}}{2}\big]\log(p-k)\frac{3\delta_1^2}{16}\big).\end{aligned}$$

This quantity tends to zero, because ε and δ_1 are fixed constants and $\sqrt{k}$ tends to infinity. We conclude that the last term in Equation (27) goes to zero, thereby concluding the proof.

3.4 Proof of Lemma 5

We first observe that conditioned on X_S, each U_i is Gaussian with mean and variance

$$\begin{aligned}m_i &:= \mathbb{E}[U_i \mid X_S] = e_i^T\big(\frac{1}{n}X_S^TX_S\big)^{-1}\big[-\rho_n\vec{1}\big], \quad \text{and} \\ \psi_i &:= \operatorname{var}[U_i \mid X_S] = \frac{1}{\gamma n}e_i^T\big(\frac{1}{n}X_S^TX_S\big)^{-1}e_i,\end{aligned}$$

respectively. Let us define the function

$$T(\gamma,k,p,\theta,t) \quad := \quad \frac{1}{\gamma}\sqrt{\max\big\{\frac{\log(t)}{\theta k\log(p-k)},\frac{\log[\theta\log(p-k)]}{\theta\log(p-k)}\big\}}, \tag{28}$$

as well as the the upper bounds

$$m^* := \rho_n(1+C\sqrt{k}T(\gamma,k,p,1,k)), \quad \text{and} \quad \psi^* := \frac{1}{\gamma n}(1+CT(\gamma,k,p,1,k)).$$

Now consider the event

$$\mathcal{T}(m^*,\psi^*) \quad := \quad \{\max_{i\in S}|m_i| \le m^* \text{ and } \max_{i\in S}|\psi_i| \le \psi^*\}.$$

Conditioning on $\mathcal{T}$ and its complement, we have

$$\begin{aligned}\mathbb{P}[(\mathcal{E}(U))^c] &= \mathbb{P}[\frac{1}{\beta^*_{\min}}\max_{i\in S}U_i| > 1] \\ &\le \mathbb{P}[\frac{1}{\beta^*_{\min}}\max_{i\in S}|U_i| > 1 \;\mid\; \mathcal{T}(m^*,\psi^*)] + \mathbb{P}[(\mathcal{T}(m^*,\psi^*))^c]. \end{aligned} \tag{29}$$

In order to upper bound $\mathbb{P}[(\mathcal{T}(m^*,\psi^*))^c]$, we first upper bound the terms $\mathbb{P}(|m_i| > m^*)$ and $\mathbb{P}(|\psi_i| > \psi^*)$, and then apply the union bound. Beginning with the mean, we have

$$\begin{aligned}|m_i| &:= \rho_n\big|e_i^T\big(\frac{1}{n}X_S^TX_S\big)^{-1}\vec{1}\big| \\ &= \rho_n\big|e_i^T\big[\big(\frac{1}{n}X_S^TX_S\big)^{-1} - I_{k,k}\big]\vec{1} + e_i^TI_{k,k}\vec{1}\big| \\ &\le \rho_n\big|e_i^T\big[\big(\frac{1}{n}X_S^TX_S\big)^{-1} - I_{k,k}\Big\}\vec{1}\big| + \rho_n.\end{aligned}$$

Our next step is to upper bound the operator norm of the matrix within curly braces, which we do by applying Lemma 10 from Section D, with the parameters $\theta = 1$ and $t = k$. We conclude there is an universal constant C such that $\|\|(\frac{1}{n}X_S^TX_S)^{-1} - I_{k,k}\|\|_2 \le CT(\gamma,k,p,1,k)$ with probability at least $1 - O(k^{-2})$. Consequently, if we define $m^* := \rho_n\big[\sqrt{k}CT(\gamma,k,p,1,k)\big] + \rho_n$, then we have the bound

$$\mathbb{P}\big[|m_i| \ge m^*\big] \quad = \quad O(1/k^2).$$

A similar argument can be used to bound each term ψ_i, thereby obtaining

$$\mathbb{P}[\big|\psi_i\big| > \psi^*] \quad = \quad O(k^{-2}).$$

Since there are k versions of each of m_i and ψ_i, the union bound implies that

$$\mathbb{P}[(\mathcal{T}(m^*,\psi^*))^c] \le 2kO(k^{-2}) = O(k^{-1}).$$

We now turn to the first term of Equation (29). Letting $Y_i \sim \mathcal{N}(0,\psi_i)$, and using $\mathcal{T}$ as shorthand for the event $\mathcal{T}(m^*,\psi^*)$, we have

$$\begin{aligned}
\mathbb{P}[\frac{1}{\beta^*_{\min}}\max_{i\in S}|U_i| > 1 \mid \mathcal{T}] &= \mathbb{E}\big\{\mathbb{P}\big[\max_{i\in S}|U_i| > \beta^*_{\min} \mid X_S, \mathcal{T}\big]\big\} \\
&\leq \mathbb{E}\big\{\mathbb{P}\big[\max_{i\in S}\big(|m_i|+|Y_i|\big) > \beta^*_{\min} \mid X_S, \mathcal{T}\big]\big\} \\
&\leq \mathbb{E}\big\{\mathbb{P}\big[m^* + \max_{i\in S}|Y_i| > \beta^*_{\min} \mid X_S, \mathcal{T}\big]\big\} \\
&= \mathbb{E}\big\{\mathbb{P}\big[\frac{1}{\beta^*_{\min}}\max_{i\in S}|Y_i| > 1 - \frac{m^*}{\beta^*_{\min}} \mid X_S, \mathcal{T}\big]\big\}.
\end{aligned}$$

For sufficiently large p and k, we have

$$\begin{aligned}
T(\gamma,k,p,1,k) &= \frac{1}{\gamma}\sqrt{\max\big\{\frac{\log(k)}{k\log(p-k)}, \frac{\log[\log(p-k)]}{\log(p-k)}\big\}} \\
&\leq \frac{1}{\gamma}\sqrt{\big\{\frac{\log[\log(p-k)]}{\log(p-k)}\big\}},
\end{aligned}$$

using the facts that $\frac{\log(k)}{k} \to 0$ and $\log[\log(p-k)] \to \infty$, so that the maximum is dominated by the second term. As a result, applying condition (9) yields that $\frac{m^*}{\beta^*_{\min}} \to 0$. Letting $Y^* \sim \mathcal{N}(0,\psi^*)$, we have

$$\begin{aligned}
\mathbb{E}\big\{\mathbb{P}\big[\frac{1}{\beta^*_{\min}}\max_{i\in S}|Y_i| > \frac{1}{2} \mid X_S, \mathcal{T}\big]\big\} &\leq \mathbb{E}\big\{k\,\mathbb{P}[|Y^*| \geq \frac{\beta^*_{\min}}{2} \mid X_S, \mathcal{T}\big]\big\} \\
&\leq 2k\,\exp\big(-\frac{[\beta^*_{\min}]^2}{8\psi^*}\big),
\end{aligned}$$

where the last inequality follows from Gaussian tail bounds (see Appendix A). It remains to verify that this final term converges to zero. Taking logarithms and ignoring constant terms, we have

$$\log(k)\big(1 - \frac{[\beta^*_{\min}]^2}{\log(k)\,8\psi^*}\big) = \log(k)\big(1 - \frac{[\beta^*_{\min}]^2\gamma n}{8\log(k)\,(1+CT(\gamma,k,p,1,k))}\big).$$

Our goal is to show that that this quantity diverges to $-\infty$. Condition (10) implies that

$$T(\gamma,k,p,1,k) = \frac{1}{\gamma}\sqrt{\max\big\{\frac{\log(k)}{k\log(p-k)}, \frac{\log\log(p-k)}{\log(p-k)}\big\}} \to 0.$$

Hence, it suffices to show that $\log k\,\big(1 - \frac{[\beta^*_{\min}]^2\gamma n}{16\log k}\big)$ diverges to $-\infty$. We have

$$\begin{aligned}
\log(k)\,\big(1 - \frac{[\beta^*_{\min}]^2\gamma n}{16\log(k)}\big) &= \log(k)\,(1 - \frac{[\beta^*_{\min}]^2}{\rho_n^2}\frac{\gamma n\rho_n^2}{16\log(k)}) \\
&= \log(k)\,(1 - \frac{[\beta^*_{\min}]^2}{\rho_n^2}\frac{\gamma n\rho_n^2}{16\log(p-k)}\frac{\log(p-k)}{\log(k)}).
\end{aligned}$$

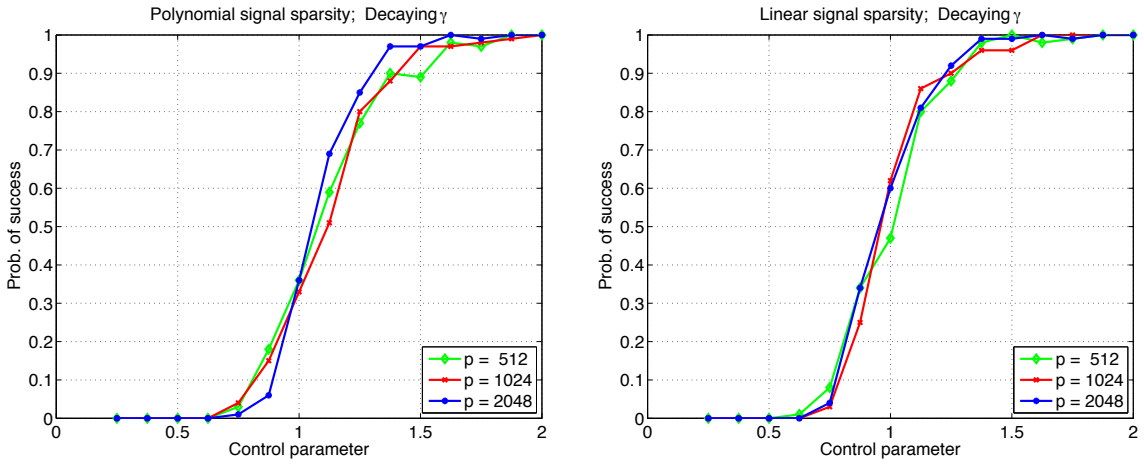
Condition (9) implies that $\frac{[\beta^*_{\min}]^2}{\rho_n^2} \to \infty$, whereas condition (8) ensures that $\frac{\gamma n\rho_n^2}{\log(p-k)} \to \infty$. The signal class $\mathcal{C}(p,k,\beta^*_{\min})$, as previously defined (1), ensures that $k \leq \frac{p}{2}$, so that the third term is greater than one. Putting together the pieces, we conclude that $\mathbb{P}[\mathcal{E}(U)^c]$ tends to zero.

4. Experimental Results

In this section, we provide some experimental results to illustrate the claims of Theorem 1. We consider two different sparsity regimes, namely linear sparsity ($k = \alpha p$) and polynomial sparsity ($k = \sqrt{p}$), and we show simulations in which the fraction γ of non-zero entries in each row of the measurement matrix X converges to zero. For all experiments, the additive noise standard deviation is set to $\sigma = 0.25$ and we fix the vector β^* by setting the first k entries are set to one, and the remaining entries to zero. There is no loss of generality in fixing the support in this way, since the ensemble of models is invariant under permutations.

Although it is possible to solve the Lasso using a variety of methods, our theory (in particular, Lemma 2) shows that it suffices to simulate the random variables $\{V_j^a, V_j^b, j \in S^c\}$ and $\{U_i, i \in S\}$, and then check the equivalent conditions (21) and (22). (These necessary and sufficient conditions give the same result for support recovery as solving the Lasso; however, they are much faster to simulate.) In all cases, we plot the success probability $\mathbb{P}[S(\widehat{\beta}) = S(\beta^*)]$ versus the *control parameter* $\theta(n,p,k) = \frac{n}{2k\log(p-k)}$. Note that Theorem 1 predicts that the Lasso should transition from failure to success at $\theta \approx 1$.

In Figure 1, the empirical success rate of the Lasso is plotted against the control parameter $\theta(n,p,k) = \frac{n}{2k\log(p-k)}$. Each panel shows three curves, corresponding to the problem sizes $p \in \{512, 1024, 2048\}$, and each point on the curve represents the average of 100 trials. For all trials shown, we set $\gamma = 0.5\frac{\log(p-k)}{\sqrt{p-k}}$, which converges to zero at a rate slightly faster than that guaranteed by Theorem 1. Nonetheless, we still observe the "stacking" behavior around the predicted threshold $\theta^* = 1$.



(a) (b)

Figure 1: Plots of the success probability $\mathbb{P}[\widehat{S} = S]$ versus the control parameter $\theta(n,p,k) = \frac{n}{k\log(p-k)}$ for γ-sparsified ensembles, with decaying measurement sparsity $\gamma = \frac{.5\log(p-k)}{\sqrt{p-k}}$. (a) Polynomial signal sparsity $k = O(\sqrt{p})$. (b) Linear signal sparsity $k = \Theta(p)$.

5. Discussion

In this paper, we have studied the problem of recovery the support set of a sparse vector β^* based on noisy observations. The main result is to show that it is possible to "sparsify" standard dense measurement matrices, so that they have a vanishing fraction of non-zeroes per row, while retaining the same sample complexity (number of observations n) required for exact recovery. We also showed that under the support recovery metric and in the presence of noise, no method can succeed without the number of non-zeroes per column tending to infinity, so that our results cannot be improved substantially. Thus, our results show that it is possible to use sparse measurement matrices while retaining the same guarantees regarding the recovery of the support. Note that our sparsification scheme is the simplest one, and requires no additional overhead to implement. Although this paper focused on sparsified Gaussian measurement matrices, it is possible to obtain qualitatively similar results for sparsified sub-Gaussian ensembles (for instance, the discrete uniform distribution on $\{-1,1\}$).

The approach taken in this paper is to find rates which γ (as a function of n, p, k) can safely tend towards zero while maintaining the same statistical efficiency as dense random matrices. In various practical settings (Wakin et al., 2006), it may be preferable to make the measurement ensembles even sparser at the cost of taking more measurements n, thereby decreasing statistical efficiency relative to dense random matrices. A natural question is the sample complexity $n(\gamma, p, k)$ in this regime as well. Finally, this work has focused only on a randomly sparsified matrices, as opposed to particular sparse designs (e.g., based on LDPC or expander-type constructions Feldman et al., 2007; Sarvotham et al., 2006; Xu and Hassibi, 2007). Although our results imply that exact support recovery with noisy observations is impossible with bounded degree designs, it would be interesting to examine the trade-off between other loss functions (e.g., ℓ_2 reconstruction error) and sparse measurement matrices.

Acknowledgments

This work was partially supported by NSF grants CAREER-CCF-0545862 and DMS-0605165, a Vodafone-US Foundation Fellowship (DO), and a Sloan Foundation Fellowship (MJW).

Appendix A. Standard Concentration Results

In this appendix, we collect some tail bounds used repeatedly throughout this paper.

Lemma 7 (Hoeffding bound—Hoeffding, 1963) *Given a binomial variate $Z \sim \mathrm{Bin}(n,\gamma)$, we have for any $\delta > 0$*

$$\mathbb{P}[|Z - \gamma n| \geq \delta n] \quad \leq \quad 2\exp\big(-2n\delta^2\big).$$

Lemma 8 (χ^2-concentration—Johnstone, 2001) *Let $X \sim \chi^2_m$ be a chi-squared variate with m degrees of freedom. Then for all $\frac{1}{2} > \delta \geq 0$, we have*

$$\mathbb{P}[X - m \geq \delta m] \quad \leq \quad \exp\big(-\frac{3}{16}m\delta^2\big).$$

We will also find the following standard Gaussian tail bound (e.g., Ledoux and Talagrand, 1991) useful:

Lemma 9 (Gaussian tail behavior) *Let $V \sim \mathcal{N}(0,\sigma^2)$ be a zero-mean Gaussian with variance σ^2. Then for all $\delta > 0$, we have*

$$\mathbb{P}[|V| > \delta] \quad \leq \quad 2\,\exp\big(-\frac{\delta^2}{2\sigma^2}\big).$$

Appendix B. Convex Optimality Conditions

In this section we discuss the optimality conditions that the Lasso must satisfy and some implications that follow.

B.1 Proof of Lemma 2

Let $f(\beta) := \frac{1}{2n}\|Y - X\beta\|_2^2 + \rho_n\|\beta\|_1$ denote the objective function of the Lasso (4). By standard convex optimality conditions (Rockafellar, 1970), a vector $\widehat{\beta} \in \mathbb{R}^p$ is a solution to the Lasso if and only if $0 \in \mathbb{R}^p$ is an element of the subdifferential of $f(\beta)$ at $\widehat{\beta}$. These conditions lead to

$$\frac{1}{n}X^T(X\widehat{\beta} - Y) + \rho_n\widehat{z} \quad = \quad 0,$$

where the dual vector $\widehat{z} \in \mathbb{R}^p$ is an element of the subdifferential of the ℓ_1-norm, given by

$$\partial\|\widehat{\beta}\|_1 \quad = \quad \big\{z \in \mathbb{R}^p \;\mid\; z_i = \mathrm{sign}(\widehat{\beta}_i) \text{ if } \widehat{\beta}_i \neq 0, \qquad z_i \in [-1,1] \text{ otherwise}\big\}.$$

Now suppose that we are given a pair $(\widehat{\beta},\widehat{z}) \in \mathbb{R}^p \times \mathbb{R}^p$ that satisfy the assumptions of Lemma 2. Condition (13) is equivalent to $(\widehat{\beta},\widehat{z})$ satisfying the zero subgradient condition. Conditions (14), (16) and (17) ensure that $\widehat{z}$ is an element of the subdifferential of the ℓ_1-norm at $\widehat{\beta}$. Finally, conditions (15) and (17) ensure that $\widehat{\beta}$ correctly specifies the signed support.

It remains to verify that $\widehat{\beta}$ is the *unique* optimal solution. By Lagrangian duality, the Lasso problem (4) (given in penalized form) can be written as an equivalent constrained optimization problem over the ball $\|\beta\|_1 \leq C(\rho_n)$, for some constant $C(\rho_n) < +\infty$. Equivalently, we can express this single ℓ_1-constraint as a set of 2^p linear constraints $\vec{v}^T\beta \leq C$, one for each sign vector $\vec{v} \in \{-1,+1\}^p$. The vector $\widehat{z}$ can be written as a convex combination $\widehat{z} = \sum_{\vec{v}} \alpha^*_{\vec{v}}\vec{v}$, where the weights $\alpha^*_{\vec{v}}$ are non-negative and sum to one. By construction of $\widehat{\beta}$ and $\widehat{z}$, the weights α^* form an optimal Lagrange multiplier vector for the problem. Consequently, any other optimal solution—say $\widetilde{\beta}$—must also minimize the associated Lagrangian

$$L(\beta;\alpha^*) \quad = \quad f(\beta) + \sum_{\vec{v}} \alpha^*_{\vec{v}}\big[\vec{v}^T\beta - C\big],$$

and satisfy the complementary slackness conditions $\alpha^*_{\vec{v}}\big(\vec{v}^T\widetilde{\beta} - C\big) = 0$ for every $\vec{v}$.

Note that these complementary slackness conditions imply that $\widehat{z}^T\widetilde{\beta} = C$. But this can only happen if $\widetilde{\beta}_j = 0$ for all indices where $|\widehat{z}_j| < 1$. Therefore, any optimal solution $\widetilde{\beta}$ satisfies $\widetilde{\beta}_{S^c} = 0$. Finally, given that all optimal solutions satisfy $\beta_{S^c} = 0$, we may consider the restricted optimization problem subject to this set of constraints. If the Hessian submatrix $\widehat{\Sigma}_{SS}$ is strictly positive definite, then this sub-problem is strictly convex, so that $\widehat{\beta}$ must be the unique optimal solution, as claimed.

B.2 Derivation of $\{V_j^a, V_j^b, U_i\}$

In this appendix, we derive the form of the $\{V_j^a, V_j^b\}$ and $\{U_i\}$ variables defined in Equations (18) through (20). We begin by writing the zero sub-gradient condition in a block-form, and substituting the relations specified in conditions (14) and (15):

$$\begin{bmatrix} \widehat{\Sigma}_{SS} & \widehat{\Sigma}_{SS^c} \\ \widehat{\Sigma}_{S^cS} & \widehat{\Sigma}_{S^cS^c} \end{bmatrix} \begin{bmatrix} \widehat{\beta}_S - \beta_S^* \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{1}{n} X_S^T W \\ \frac{1}{n} X_{S^c}^T W \end{bmatrix} + \rho_n \begin{bmatrix} \text{sign}(\beta_S^*) \\ \widehat{z}_{S^c} \end{bmatrix} \quad = \quad 0.$$

By solving the top block, we obtain

$$U := \widehat{\beta}_S - \beta_S^* \quad = \quad \widehat{\Sigma}_{SS}^{-1} \big\{ \frac{1}{n} X_S^T W - \rho_n \, \text{sign}(\beta_S^*) \big\}.$$

By back-substituting this relation into the lower block, we can solve explicitly for $\widehat{z}_{S^c}$; doing so yields that $\rho_n \widehat{z}_{S^c} = V^a + V^b$, where the $(p-k)$-vectors are defined in Equations (18) and (19).

Appendix C. Proof of Lemma 6

Let $Z \in \mathbb{R}^{n \times n}$ denote a $n \times n$ matrix, for which the off-diagonal elements $Z_{ij} = 0$ for all $i \neq j$, and the diagonal elements $Z_{ii} \sim \text{Ber}(\gamma)$ are i.i.d. With this notation, we can write $H \stackrel{d}{=} Zh$. Using the definition (23) of h, we have

$$\begin{aligned} \| H \|_2^2 &= \|Zh\|_2^2 \\ &= \|Z \frac{X_S}{n} (\widehat{\Sigma}_{SS})^{-1} \vec{1} \|_2^2 \\ &= \vec{1}^T (\widehat{\Sigma}_{SS})^{-1} (Z\frac{X_S}{n})^T (Z\frac{X_S}{n}) (\widehat{\Sigma}_{SS})^{-1} \vec{1} \\ &= \frac{\gamma}{n} \vec{1}^T (\widehat{\Sigma}_{SS})^{-1} \underbrace{\big\{ \frac{1}{\gamma n} \sum_{i=1}^n \mathbb{I}[Z_{ii} = 1] \, x_i x_i^T \big\}}_{\Gamma(Z)} (\widehat{\Sigma}_{SS})^{-1} \vec{1}, \end{aligned}$$

where x_i is the i^{th} row of the matrix X_S. We can apply Lemma 10 from Appendix D with parameters $\theta = 1$ and $t = (p-k)$ yielding

$$\mathbb{P}\big[|||\widehat{\Sigma}_{SS}^{-1}|||_2 \geq f_1(p,k,\gamma) \big] \quad \leq \quad O\big(\frac{1}{(p-k)^2}\big), \tag{30}$$

where $f_1(p,k,\gamma) := 1 + T(\gamma, k, p, 1, p-k)$, and the function $T(\gamma, k, p, \theta, t)$ was defined in Equation (28).

Next we control the spectral norm of the random matrix $\Gamma(Z)$, conditioned on the total number $\sum_{i=1}^n Z_{ii}$ of non-zero entries. In particular, applying Lemma 10 with $t = p-k$, and $\theta = \frac{z}{n}$, we have

$$\mathbb{P}\big[\| \Gamma(Z) \|_2 \geq \frac{z}{n\gamma} \big[1 + T(\gamma, k, p, \frac{z}{n}, p-k) \big] \; \big| \; \sum_{i=1}^n Z_{ii} = z \big] \quad \leq \quad \frac{1}{(p-k)^2}, \tag{31}$$

as long as $k\frac{z}{n} \to \infty$.

The next step is to deal with the conditioning. Define the event

$$\mathcal{T}(k,\gamma) \;:=\; \big\{Z \mid \gamma - \frac{1}{2\sqrt{k}} \leq \frac{1}{n}\sum_{i=1}^{n} Z_{ii} \leq \gamma + \frac{1}{2\sqrt{k}}\big\}.$$

We need to find an upper bound on $\|\Gamma(Z)\|_2$ that will hold with high probability for all Z that satisfy the above property. One function that suffices is

$$f_2(p,k,\gamma) \;:=\; \left(1+\frac{1}{2\sqrt{k}\gamma}\right)\left[1+C\Big(\frac{1}{\gamma}\sqrt{\max\left\{\frac{1}{k(\gamma-\frac{1}{2\sqrt{k}})}, \frac{\log[(\gamma+\frac{1}{2\sqrt{k}})\log(p-k)]}{(\gamma-\frac{1}{2\sqrt{k}})\log(p-k)}\right\}}\Big)\right].$$

We then have that

$$\begin{aligned}
\mathbb{P}[|||\Gamma(Z)|||_2 \geq f_2(p,k,\gamma)] &\leq \mathbb{P}[|||\Gamma(Z)|||_2 \geq f_2(p,k,\gamma) \mid \mathcal{T}(k,\gamma)] + \mathbb{P}[(\mathcal{T}(k,\gamma))^c] \\
&\leq \exp(-2\log(p-k)) + 2\exp(-\frac{n}{2k}) \\
&\leq 3\exp(-\min\{2\log(p-k), \frac{n}{2k}\}), \qquad (32)
\end{aligned}$$

where we have used the bound (31), and the Hoeffding bound (see Lemma 7).

Combining the bounds (30) and (32), we conclude that as long as $\gamma k \to \infty$, then:

$$\mathbb{P}\big[|||\widehat{\Sigma}^{-1}\Gamma(Z)\widehat{\Sigma}^{-1}|||_2 \geq f_1^2 f_2\big] \;\leq\; 4\exp(-\min\{2\log(p-k), \frac{n}{2k}\}).$$

Since $\|\vec{1}\|_2 = \sqrt{k}$, we have

$$\mathbb{P}[\|H\|_2^2 \geq \frac{\gamma k}{n} f_1^2 f_2] \;\leq\; 4\exp(-\min\{2\log(p-k), \frac{n}{2k}\}).$$

To conclude the proof, we must show that both $f_1(p,k,\gamma)$ and $f_2(p,k,\gamma)$ converge to 1 as (p,k,γ) scale. The term $f_1(p,k,\gamma) = 1 + T(\gamma,k,p,1,p-k)$ converges to one, since the quantity

$$T(\gamma,k,p,1,p-k) \;=\; \frac{1}{\gamma}\sqrt{\max\big\{\frac{1}{k}, \frac{\log[\log(p-k)]}{\log(p-k)}\big\}}$$

converges[1] to zero under assumption (10). Next, we need to demonstrate that $f_2(p,k,\gamma)$ converge to 1 as (p,k,γ) scale. Since assumption (10) ensures that $\gamma\sqrt{k} \to \infty$, it suffices to study the simpler function

$$f_3(p,k,\gamma) \;:=\; 1 + C\Big(\frac{1}{\gamma}\sqrt{\max\left\{\frac{1}{k\gamma}, \frac{\log[\gamma\log(p-k)]}{\gamma\log(p-k)}\right\}}\Big),$$

which has the same asymptotic behavior as $f_2(p,k,\gamma)$. Observe that $f_3(p,k,\gamma)$ satisfies the sandwich relation

$$1 \;\leq\; f_3(p,k,\gamma) \;\leq\; 1 + C\Big(\sqrt{\max\big\{\frac{1}{k\gamma^3}, \frac{\log[\log(p-k)]}{\gamma^3\log(p-k)}\big\}}\Big),$$

By assumption (10), this upper bound converges to one, showing that f_3 and hence f_2 converge to one, as desired. In particular, for any fixed $\delta > 0$, we have $f_1^2 f_2 < (1+\delta)$ for p,k sufficiently large, thereby completing the proof of Lemma 6.

1. In particular, the left-hand side of the expression (10) satisfies $\frac{\gamma}{[T(\gamma,k,p,1,p-k)]^2} \leq \frac{1}{[T(\gamma,k,p,1,p-k)]^2}$.

Appendix D. Singular Values of Sparsified Matrices

Let $\theta(p,k) \in (0,1]$ and $t(p,k) \in \{1,2,3,\ldots\}$ be functions. Let X be an $\theta n \times k$ random matrix with i.i.d. entries X_{ij} distributed according to the γ-sparsified ensemble (6). Recall the definition of the function $T(\gamma,k,p,\theta,t)$ defined in Equation (28), and let $t > 0$ be arbitrary.

Lemma 10 *Suppose that $n \geq (2+\nu)k\log(p-k)$ for some $\nu > 0$. If as k and $p-k \to \infty$, we have $T(\gamma,k,p,\theta,t) \longrightarrow 0$, then there are universal positive constants C_i such that*

$$\mathbb{P}\big[\sup_{\|u\|_2=1} \big|\frac{1}{\sqrt{\theta n}}\|Xu\|_2 - 1\big| \geq C_1 T(\gamma,k,p,\theta,t)\big] \;=\; O(\frac{1}{t^2}), \quad \textit{and} \tag{33}$$

$$\mathbb{P}\big[\big|\|(\frac{1}{\theta n}X^TX)^{-1} - I_{k\times k}\|_2 \geq C_2 T(\gamma,k,p,\theta,t)\big] \;=\; O(\frac{1}{t^2}). \tag{34}$$

Remark 3 *(a) Note that Equation* (33) *implies that the eigenvalues of the matrix $\frac{1}{\theta n}X^TX$ are contained in the interval $(1-C_1T(\gamma,k,p,\theta,t), 1+C_1T(\gamma,k,p,\theta,t))$. Since $T(\gamma,k,p,\theta,t) = o(1)$ by assumption, we can always find a constant C_2 such that the eigenvalues of the inverted matrix $(\frac{1}{\theta n}X^TX)^{-1}$ are contained in the interval $(1-C_2T(\gamma,k,p,\theta,t), 1+C_2T(\gamma,k,p,\theta,t))$. Consequently, Equation* (34) *is a consequence of the assumptions of Lemma 10 and Equation* (33).

(b) In addition, observe that Lemma 10 with $\theta = 1$ and $t = p-k$ implies that $\widehat{\Sigma} = \frac{1}{n}X_S^TX_S$ is invertible with probability greater than $1 - O(\frac{1}{(p-k)^2})$, there establishing Lemma 1. Other settings in which this lemma is applied are $(\theta,t) = (\gamma, p-k)$ and $(\theta,t) = (1,k)$.

The remainder of this section is devoted to the proof of Lemma 10.

D.1 Bounds on Expected Values

Let $X \in \mathbb{R}^{\theta n \times k}$ be a random matrix with i.i.d. entries from the γ-sparsified ensemble

$$X_{ij} \;\sim\; (1-\gamma)\delta_X(0) + \gamma\mathcal{N}(0,\frac{1}{\gamma}).$$

Note that $\mathbb{E}[X_{ij}] = 0$ and $\mathrm{var}(X_{ij}) = 1$ by construction.

We follow the proof technique outlined in the lecture notes (Vershynin, 2006). We first note the tail bound:

Lemma 11 *Let $Q_1,\ldots,Q_d$ be i.i.d. samples of the γ-sparsified Gaussian ensemble. Given any vector $a \in \mathbb{R}^d$ and $t > 0$, we have $\mathbb{P}[\sum_{i=1}^d a_iQ_i > t] \leq \exp\big(-\frac{\gamma t^2}{2\|a\|_2^2}\big)$.*

To establish this bound, note that each Y_i is dominated (stochastically) by the random variable $Z \sim \mathcal{N}(0,\frac{1}{\gamma})$. In particular, for any $\lambda > 0$ we have

$$\mathbb{M}_{Q_i}(\lambda) \;=\; \mathbb{E}[\exp(\lambda Q_i)] \;=\; (1-\gamma) + \gamma\mathbb{E}[\exp(\lambda Z)] \leq \exp(\lambda^2/2\gamma),$$

from which the claim follows by optimizing the Chernoff bound.

Now let us bound the maximum singular value $s_{\max}(X)$ of the random matrix X. Letting S^{d-1} denote the ℓ_2 unit ball in d dimensions, we begin with the variational representation

$$\begin{aligned} s_{\max}(X) &= \max_{u \in S^{k-1}} \|Xu\| \\ &= \max_{v \in S^{\theta n-1}} \max_{u \in S^{k-1}} v^T X u. \end{aligned}$$

For an arbitrary $\varepsilon \in (0,1)$, we can find ε-covers (in ℓ_2 norm) of $S^{\theta n-1}$ and S^{k-1} with $M_{\theta n}(\varepsilon) = (3/\varepsilon)^{\theta n}$ and $M_k(\varepsilon) = (3/\varepsilon)^k$ points respectively (Matousek, 2002). Denote these covers by $C_{\theta n}(\varepsilon)$ and $C_k(\varepsilon)$ respectively. A standard argument shows that for all $\varepsilon \in (0,1)$, we have

$$\|X\|_2 \leq \frac{1}{(1-\varepsilon)^2} \max_{u_\alpha \in C_k(\varepsilon)} \max_{v_\beta \in C_{\theta n}(\varepsilon)} v_\beta^T X u_\alpha.$$

Let us analyze the maximum on the RHS: for a fixed pair (u,v) in our covers, we have

$$u^T X v = \sum_{i=1}^{k} \sum_{j=1}^{\theta n} X_{ij} u_i v_j.$$

Let us apply Lemma 11 with $d = \theta n k$, and weights $a_{ij} = u_i v_j$. Note that we have

$$\|a\|_2^2 = \sum_{i,j} a_{ij}^2 = \sum_i u_i^2 (\sum_j v_j^2) = 1,$$

since each u and v are unit norm. Consequently, for any fixed u, v in the covers, we have

$$\mathbb{P}[u^T X v > t] \leq \exp\big(-\frac{\gamma t^2}{2}\big).$$

By the union bound, we have

$$\begin{aligned} \mathbb{P}\big[\max_{u_\alpha \in C_k(\varepsilon)} \max_{v_\beta \in C_{\theta n}(\varepsilon)} v_\beta^T X u_\alpha > t\big] &\leq M_k(\varepsilon) M_{\theta n}(\varepsilon) \exp\big(-\frac{\gamma t^2}{2}\big) \\ &\leq \exp\big((k+\theta n)\log(3/\varepsilon) - \frac{\gamma t^2}{2}\big). \end{aligned}$$

By choosing $\varepsilon = \frac{1}{2}$ and $t = \sqrt{\frac{4}{\gamma}(k+\theta n)\log 6}$, we can conclude that

$$s_{\max}(X)/\sqrt{\theta n} = \|X\|_2/\sqrt{\theta n} \leq C\sqrt{\frac{1}{\gamma}}\sqrt{1+\frac{k}{\theta n}},$$

with probability at least $1 - \exp(-(k+\theta n)\log 6)$. Note that

$$\frac{k}{\theta n} = O\big(\frac{1}{(2+\nu)\theta \log(p-k)}\big) \to 0,$$

since $\frac{\log[\theta\log(p-k)]}{\theta\log(p-k)} \to 0$, which implies that $\theta \log(p-k) \to \infty$.

Consequently, we can conclude that

$$\|X\|_2/\sqrt{\theta n} \leq O(1/\sqrt{\gamma}),$$

w.p. one as $\theta n, k \to \infty$. Although this bound is essentially correct for a $\mathcal{N}(0, \frac{1}{\gamma})$ ensemble with γ *fixed*, it is very crude for the sparsified case with $\gamma \to 0$, but will useful in obtaining tighter control on $s_{\max}(X)$ and $s_{\min}(X) := \min_{u \in S^{k-1}} \|Xu\|$ in the sequel.

D.2 Tightening the Bound

For a given $u \in S^{k-1}$, consider the random variable $\|Xu\|_2^2 := \sum_{i=1}^{\theta n}(Xu)_i^2$. We first claim that each variate $Z_i = (Xu)_i^2$ is subexponential, or more precisely:

Lemma 12 *For any $s > 0$, we have $\mathbb{P}[Z_i > s] \leq 2\exp\big(-\frac{\gamma s}{2}\big)$.*

Proof We can write $(Xu)_i = \sum_{j=1}^k X_{ij}u_j$ where $\|u\|_2 = 1$. Consequently, Lemma 11 implies that $\mathbb{P}[\sum_{j=1}^k X_{ij}u_j > \delta] \leq \exp(-\frac{\gamma\delta^2}{2})$. By symmetry, we have

$$\mathbb{P}[Z_i > s] = \mathbb{P}[|\sum_{j=1}^k X_{ij}u_j| > \sqrt{s}] \;\leq\; 2\exp(-\frac{\gamma s}{2}),$$

from which the claim follows. ■

Now consider the event

$$\big\{\big|\frac{\|Xu\|_2^2}{\theta n} - 1\big| > \delta\big\} \quad = \quad \big\{\big|\sum_{i=1}^{\theta n} Z_i - \mathbb{E}[\sum_{i=1}^{\theta n} Z_i]\big| > \delta\theta n\big\}.$$

Let us apply Theorem 1.4 from Vershynin (2000) with $a = 2$, $b = 8\theta n/\gamma^2$ and $d = 2/\gamma$. With these choices, we have $4b/d = 16\theta n/\gamma$, which grows at least linearly in θn. Consequently, for any $\delta > 0$ less than $16/\gamma$ (we will in fact take $\delta \to 0$), we have

$$\mathbb{P}\big[\big|\frac{\|Xu\|_2^2}{\theta n} - 1\big| > \delta\big] \quad \leq \quad 2\exp\big(-\frac{\delta^2(\theta n)^2}{256\theta n/\gamma^2}\big) \;=\; 2\exp\big(-\frac{\gamma^2\,\delta^2\theta n}{256}\big).$$

Now take an ε-cover of the k-dimensional ℓ_2 ball, say with $N(\varepsilon) \;=\; (3/\varepsilon)^k$ elements. By the union bound, we have

$$\mathbb{P}\big[\inf_{i=1,\ldots,N(\varepsilon)} \frac{\|Xu_i\|_2^2}{\theta n} < 1-\delta\big] \quad \leq \quad \exp\big(-\frac{\gamma^2\,\delta^2\theta n}{256} + k\log(3/\varepsilon)\big).$$

Now set

$$\delta = \frac{\sqrt{2}}{\gamma}\sqrt{\frac{256 f(k,p) k\log(3/\varepsilon)}{\theta n}},$$

where $f(k,p) \geq 1$ is a function to be specified. Doing so yields that the infimum is bounded by $1+\delta$ with probability $1-\exp(-kf(k,p)\log(3/\varepsilon))$. (Note that the choice of $f(k,p)$ influences the rate of convergence, hence its utility.)

For any element $u \in S^{k-1}$, we have some u_i in the cover, and moreover

$$\begin{aligned}\big|\|Xu\|^2 - \|Xu_i\|^2\big| \quad &= \quad \big|\big\{\|Xu\| - \|Xu_i\|\big\}\,\big\{\|Xu\| + \|Xu_i\|\big\}\big| \\ &\leq \quad \big|\big\{\|Xu\| - \|Xu_i\|\big\}\big|\,(2\|X\|) \\ &\leq \quad (\|X\|\,\|u-u_i\|)\;(2\|X\|) \;\leq\; 2\|X\|^2\varepsilon.\end{aligned}$$

From our earlier result, we know that $\|X\|^2 = O(\theta n/\gamma)$ with probability $1-\exp(\log 6(k+\theta n))$. Putting together the pieces, we have there is a universal constant $C_2 > 0$, independent of $((\theta n), k, \gamma)$, such that the bound

$$\frac{1}{\theta n} \inf_{u \in S^{k-1}} \|Xu\|^2 \;\geq\; 1-\delta - C_3\varepsilon/\gamma \;=\; 1 - \frac{2}{\gamma}\sqrt{\frac{32 f(k,p) k \log(3/\varepsilon)}{\theta n}} - \frac{C_2}{\gamma}\varepsilon,$$

holds with probability at least

$$1 - \exp(-k f(k,p)\log(3/\varepsilon)) - \exp(-\log 6(k+\theta n)). \tag{35}$$

Setting $\varepsilon = 3k/\theta n$ yields the bound

$$\frac{1}{\theta n} \inf_{u \in S^{k-1}} \|Xu\|^2 \;\geq\; 1 - \frac{C_3}{\gamma}\sqrt{f(k,p)\frac{k}{\theta n}\log(\frac{\theta n}{k})},$$

where we have used the fact that $\frac{k}{\theta n} = o\big(\sqrt{f(k,p)\frac{k}{\theta n}\log(\frac{\theta n}{k})}\big)$. To understand how to choose the function $f(k,p)$, let us consider the rate of convergence (35). To establish the claim (33), we need rates fast enough to dominate a $2\log(t)$ term in the exponent, which guides our choice of $f(k,p)$. Recall that we are seeking to prove a scaling of the form $n = \Theta(k\log(p-k))$, so that this requirement (with $\varepsilon = 3k/\theta n = \frac{3}{\theta\log(p-k)}$) is equivalent to the quantity

$$k f(k,p)\log(3/\varepsilon) - 2\log(t) \;=\; k f(k,p)\log[\theta\log(p-k)] - 2\log(t) \tag{36}$$

tending to infinity.

D.2.1 CASE 1

If $k > \frac{\log(t)}{\log[\theta\log(p-k)]}$, then we may set $f(k,p) = 4$. With this choice, the condition (36) is satisfied, and we have

$$f(k,p)\frac{k}{\theta n}\log(\frac{\theta n}{k}) \;=\; 4\frac{\log[\theta\log(p-k)]}{\theta\log(p-k)} \;\to\; 0,$$

where we have used the assumption that $T(\gamma,k,p,\theta,t) = o(1)$.

D.2.2 CASE 2

Otherwise, if $k \leq \frac{\log(t)}{\log\theta\log(p-k)}$, then we may set

$$f(k,p) = 4\frac{\log(t)}{k\log\theta\log(p-k)} \geq 4,$$

so that condition (36) is satisfied, and we have

$$f(k,p)\frac{k}{\theta n}\log(\frac{\theta n}{k}) \;\leq\; 4\frac{\log(t)}{k\log\theta\log(p-k)}\frac{1}{\theta\log(p-k)}\log\theta\log(p-k) \;=\; \frac{4}{k}\frac{\log t}{\theta\log(p-k)} \to 0,$$

where we have again used the assumption $T(\gamma,k,p,\theta,t) = o(1)$ from Lemma 10.

Recalling the definition of $T(\gamma,k,p,\theta,t)$ from Equation (28), we can summarize both cases cleanly as

$$\mathbb{P}\left[\frac{1}{\theta n}\inf_{u\in S^{k-1}}\|Xu\|^2 \leq 1-CT(\gamma,k,p,\theta,t)\right] \quad = \quad O(1/t^2).$$

For (p,k) sufficiently large, we have $CT(\gamma,k,p,\theta,t)<1$, so that we can take square roots. Using the expansion $\sqrt{1+x}=1+\frac{x}{2}+o(x)$ for x small, we conclude that

$$\frac{1}{\sqrt{\theta n}}\inf_{u\in S^{k-1}}\|Xu\| \quad \geq \quad 1-\frac{C}{2}T(\gamma,k,p,\theta,t)-o(T(\gamma,k,p,\theta,t)),$$

with probability greater than $1-O(1/t^2)$. For (k,p) sufficiently large, the second term is smaller than $\frac{C}{4}T(\gamma,k,p,\theta,t)$, so that we conclude that

$$\mathbb{P}\left[\frac{1}{\sqrt{\theta n}}\inf_{u\in S^{k-1}}\|Xu\| \geq 1-\frac{3C}{4}T(\gamma,k,p,\theta,t)\right] \quad \geq \quad 1-O(1/t^2)$$

for (k,p) sufficiently large.

This same process can be repeated to bound the maximum singular value, yielding the bound

$$\mathbb{P}\left[\frac{1}{\sqrt{\theta n}}\sup_{u\in S^{k-1}}\|Xu\| \leq 1+\frac{3C}{4}T(\gamma,k,p,\theta,t)\right] \quad \geq \quad 1-O(1/t^2)$$

for (k,p) sufficiently large. Combining these two bounds yields the claim of Lemma 10.

References

D. Achlioptas. Database-friendly random projections. In *Proc. ACM Symp. Princ. Database Systems (PODS)*, pages 274–281, New York, USA, 2001. ACM.

N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29, New York, NY, USA, 1996. ACM Press.

R. Baraniuk, M. Davenport, R. Devore, and M. Wakin. A simple proof of the restricted isometry property for random matrices. *Constr. Approx*, 2007.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, UK, 2004.

E. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. Info Theory*, 51(12):4203–4215, December 2005.

S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Computing*, 20(1):33–61, 1998.

G. Cormode and S. Muthukrishnan. Towards an algorithmic theory of compressed sensing. Technical report, Rutgers University, July 2005.

S. Dasgupta. Learning mixtures of Gaussians. In *IEEE Symp. Foundations of Computer Science (FOCS)*, September 1999.

S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22(1):60–65, 2003.

D. Donoho. For most large underdetermined systems of linear equations, the minimal ℓ_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6): 797–829, June 2006.

J. Feldman, T. Malkin, R. A. Servedio, C. Stein, and M. J. Wainwright. LP decoding corrects a constant fraction of errors. *IEEE Trans. Information Theory*, 53(1):82–89, January 2007.

A. Gilbert, M. Strauss, J. Tropp, and R. Vershynin. Algorithmic linear dimension reduction in the ℓ_1-norm for sparse vectors. In *Proc. Allerton Conference on Communication, Control and Computing*, Allerton, IL, September 2006.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, May 2006.

W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mapping into hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

I. M. Johnstone. Chi-square oracle inequalities. In M. de Gunst, C. Klaassen, and A. van der Vaart, editors, *State of the Art in Probability and Statistics*, number 37 in IMS Lecture Notes, pages 399–418. Institute of Mathematical Statistics, 2001.

I. M. Johnstone and A. Lu. On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, 104(486):682–693, 2009.

M. Ledoux and M. Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer-Verlag, New York, NY, 1991.

P. Li, T. J. Hastie, and K. W. Church. Very sparse random projections. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296, New York, NY, USA, 2006. ACM.

P. Li, T. J. Hastie, and K. W. Church. Nonlinear estimators and tail bounds for dimension reduction in l1 using cauchy random projections. *Journal of Machine Learning Research*, 8:2497–2532, 2007.

J. Matousek. *Lectures on Discrete Geometry*. Springer-Verlag, New York, 2002.

N. Meinshausen and P. Buhlmann. High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34:1436–1462, 2006.

D. Paul. Asymptotics of sample eigenstructure for a large-dimensional spiked covariance model. *Statistica Sinica*, 17:1617–1642, 2007.

P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional ising model selection using ℓ_1-regularized logistic regression. *Annals of Statistics*, 38(3):1287–1319, 2010.

G. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, 1970.

S. Sarvotham, D. Baron, and R. G. Baraniuk. Sudocodes: Fast measurement and reconstruction of sparse signals. In *Int. Symposium on Information Theory*, Seattle, WA, July 2006.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

J. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Trans. Info Theory*, 52(3):1030–1051, March 2006.

R. Vershynin. On large random almost euclidean bases. *Acta. Math. Univ. Comenianae*, LXIX: 137–144, 2000.

R. Vershynin. Random matrix theory. Technical report, UC Davis, 2006. Lecture Notes.

M. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (Lasso). *IEEE Trans. Inf. Theor.*, 55(5):2183–2202, 2009.

M. B. Wakin, J. N. Laska, M. F. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. F. Kelly, and R. G. Baraniuk. An architecture for compressive imaging. *IEEE Int. Conf. Image Proc.*, 2006.

W. Wang, M. Garofalakis, and K. Ramchandran. Distributed sparse random projections for refinable approximation. In *Proc. International Conference on Information Processing in Sensor Networks*, Nashville, TN, April 2007.

W. Wang, M.J. Wainwright, and K. Ramchandran. Information-theoretic limits on sparse signal recovery: Dense versus sparse measurement matrices. *IEEE Trans. Inf. Theor.*, 56(6):2967–2979, 2010.

W. Xu and B. Hassibi. Efficient compressive sensing with deterministic guarantees using expander graphs. *Information Theory Workshop, 2007. ITW '07. IEEE*, 2007.

P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

S. Zhou, J. Lafferty, and L. Wasserman. Compressed regression. In *Neural Information Processing Systems*, December 2007.

Journal of Machine Learning Research 11 (2010) 2387-2422 Submitted 12/09; Revised 6/10; Published 9/10

Composite Binary Losses

Mark D. Reid* MARK.REID@ANU.EDU.AU
Robert C. Williamson* BOB.WILLIAMSON@ANU.EDU.AU
School of Computer Science, Building 108
Australian National University
Canberra ACT 0200, Australia

Editor: Rocco Servedio

Abstract

We study losses for binary classification and class probability estimation and extend the understanding of them from margin losses to general composite losses which are the composition of a proper loss with a link function. We characterise when margin losses can be proper composite losses, explicitly show how to determine a symmetric loss in full from half of one of its partial losses, introduce an intrinsic parametrisation of composite binary losses and give a complete characterisation of the relationship between proper losses and "classification calibrated" losses. We also consider the question of the "best" surrogate binary loss. We introduce a precise notion of "best" and show there exist situations where two convex surrogate losses are incommensurable. We provide a complete explicit characterisation of the convexity of composite binary losses in terms of the link function and the weight function associated with the proper loss which make up the composite loss. This characterisation suggests new ways of "surrogate tuning" as well as providing an explicit characterisation of when Bregman divergences on the unit interval are convex in their second argument. Finally, in an appendix we present some new algorithm-independent results on the relationship between properness, convexity and robustness to misclassification noise for binary losses and show that all convex proper losses are non-robust to misclassification noise.

Keywords: surrogate loss, convexity, probability estimation, classification, Fisher consistency, classification-calibrated, regret bound, proper scoring rule, Bregman divergence, robustness, misclassification noise

1. Introduction

A *loss* function is the means by which a learning algorithm's performance is judged. A *binary* loss function is a loss for a supervised prediction problem where there are two possible labels associated with the examples. A *composite* loss is the composition of a proper loss (defined below) and a link function (also defined below). In this paper we study composite binary losses and develop a number of new characterisation results. Several of these results can be seen as an extension of the work by Buja et al. (2005) applied to an analysis of composite losses by Masnadi-Shirazi and Vasconcelos (2009).

Informally, proper losses are well-calibrated losses for class probability estimation, that is for the problem of not only predicting a binary classification label, but providing an estimate of the probability that an example will have a positive label. Link functions are often used to map the outputs of a predictor to the interval $[0,1]$ so that they can be interpreted as probabilities. Having such

*. Also at National ICT Australia.

probabilities is often important in applications, and there has been considerable interest in understanding how to get accurate probability estimates (Platt, 2000; Gneiting and Raftery, 2007; Cohen and Goldszmidt, 2004) and understanding the implications of requiring loss functions provide good probability estimates (Bartlett and Tewari, 2007).

Much previous work in the machine learning literature has focussed on *margin losses* which intrinsically treat positive and negative classes symmetrically. However it is now well understood how important it is to be able to deal with the non-symmetric case (Zellner, 1986; Elkan, 2001; Provost and Fawcett, 2001; Buja et al., 2005; Bach et al., 2006; Beygelzimer et al., 2008; Christoffersen and Diebold, 2009). A key goal of the present work is to consider composite losses in the general (non-symmetric) situation. Since our development is for completely general losses, we automatically cover non-symmetric losses. The generalised notion of classification calibration developed in §5 is intrinsically non-symmetric.

1.1 Overview and Contributions

We now provide an overview of the paper's structure, highlighting the novel contributions and how they relate to existing work. Central to this work are the notions of a loss and its associated conditional and full risk. These are introduced and briefly discussed in §2.

In §3 we introduce losses for Class Probability Estimation (CPE), define some technical properties of them, and present some structural results originally by Shuford et al. (1966) and Savage (1971) and recently studied in a machine learning context by Buja et al. (2005) and Masnadi-Shirazi and Vasconcelos (2009). The most important of these are Theorem 4 which gives a representation of proper losses in terms of its associated conditional Bayes risk function, and Theorem 1 which relates a proper loss's partial losses to its "weight function"—the negative second derivative of the conditional Bayes risk (see Corollary 3). We use these to provide a novel characterisation of proper symmetric CPE losses. Specifically, Theorem 9 shows these losses are completely determined by the behaviour of one of its partial losses on half the unit interval.

Learning algorithms often make real-valued predictions that are not directly interpretable as probability estimates but require a link function which maps their output to the interval $[0,1]$. In §4 we define composite losses as the composition of a CPE loss and a link. The new contributions of this section are Theorem 10 which generalises Theorem 1 to composite losses, and Corollaries 12 and 14 which shows how requiring properness completely determines the link function for composite and margin losses. We also introduce a natural and intrinsic parametrisation of proper composite losses that is a generalisation of the weight function and show how it can be used to easily derive gradients for stochastic descent algorithms.

In §5 we generalise the notion of classification calibrated losses (as studied, for example, by Bartlett et al., 2006) so it applies to non-symmetric composite losses (i.e., not just margin losses) and provide a characterisation of it in Theorem 17. We also describe how this new notion of classification calibrated relates to proper CPE and composite losses via its connection with the weight function.

The main results of this paper are found in §6: Theorems 24 and 29 characterise when proper composite losses are convex. These characterisation are in terms of some easily testable constraints relating the losses' weight and link functions. The results also characterise when a Bregman divergence on $[0,1]$ is convex in its second argument (§6.3).

In §7 we study how the above insights can be applied to the problem of choosing a surrogate loss. Here, a *surrogate* loss function is a loss function which is not exactly what one wishes to minimise but is easier to work with algorithmically. This is still a relatively new area of research and our aim here is to open up a discussion rather than have the final word. To do so we define a well founded notion of "best" surrogate loss and show that some convex surrogate losses are incommensurable on some problems. We also consider some other notions of "best" and explicitly determine the surrogate loss that has the best surrogate regret bound in a certain sense.

Finally, in §8 we draw some more general conclusions. In particular, we argue that the weight and link function parametrisation of losses provides a convenient way to work with an entire class of losses that are central to probability estimation and may provide new ways of approaching the problem of "surrogate tuning" (Nock and Nielsen, 2009b).

Appendix C collects several observations which build upon some of the results in the main paper but are digressions from its central themes. In it, we present some new algorithm-independent results on the relationship between properness, convexity and robustness to misclassification noise for binary losses and show that all convex proper losses are non-robust to misclassification noise.

2. Losses and Risks

We write $x \wedge y := \min(x, y)$ and $[\![p]\!] = 1$ if p is true and $[\![p]\!] = 0$ otherwise.[1] The generalised function $\delta(\cdot)$ is defined by $\int_a^b \delta(x) f(x) dx = f(0)$ when f is continuous at 0 and $a < 0 < b$. Random variables are written in sans-serif font: X, Y.

Given a set of labels $\mathcal{Y} := \{-1, 1\}$ and a set of prediction values $\mathcal{V}$ we will say a *loss* is any function[2] $\ell : \mathcal{Y} \times \mathcal{V} \to [0, \infty)$. We interpret such a loss as giving a penalty $\ell(y, v)$ when predicting the value v when an observed label is y. We can always write an arbitrary loss in terms of its *partial losses* $\ell_1 := \ell(1, \cdot)$ and $\ell_{-1} := \ell(-1, \cdot)$ using

$$\ell(y, v) = [\![y = 1]\!] \ell_1(v) + [\![y = -1]\!] \ell_{-1}(v).$$

Our definition of a loss function covers all commonly used *margin losses* (i.e., those which can be expressed as $\ell(y, v) = \phi(yv)$ for some function $\phi : \mathbb{R} \to [0, \infty)$) such as the *0-1 loss* $\ell(y, v) = [\![yv < 0]\!]$, the *hinge loss* $\ell(y, v) = \max(1 - yv, 0)$, the *logistic loss* $\ell(y, v) = \log(1 + e^{yv})$, and the *exponential loss* $\ell(y, v) = e^{-yv}$ commonly used in boosting. It also covers *class probability estimation losses* where the predicted values $\hat{\eta} \in \mathcal{V} = [0, 1]$ are directly interpreted as probability estimates.[3] We will use $\hat{\eta}$ instead of v as an argument to indicate losses for class probability estimation and use the shorthand *CPE losses* to distinguish them from general losses. For example, *square loss* has partial losses $\ell_{-1}(\hat{\eta}) = \hat{\eta}^2$ and $\ell_1(\hat{\eta}) = (1 - \hat{\eta})^2$, the *log loss* $\ell_{-1}(\hat{\eta}) = \log(1 - \hat{\eta})$ and $\ell_1(\hat{\eta}) = \log(\hat{\eta})$, and the family of *cost-weighted misclassification losses* parametrised by $c \in (0, 1)$ is given by

$$\ell_c(-1, \hat{\eta}) = c [\![\hat{\eta} \geq c]\!] \text{ and } \ell_c(1, \hat{\eta}) = (1 - c) [\![\hat{\eta} < c]\!]. \tag{1}$$

1. This is the Iverson bracket notation as recommended by Knuth (1992).
2. Restricting the output of a loss to $[0, \infty)$ is equivalent to assuming the loss has a lower bound and then translating its output.
3. These are known as *scoring rules* in the statistical literature (Gneiting and Raftery, 2007).

2.1 Conditional and Full Risks

Suppose we have random examples X with associated labels $\mathsf{Y} \in \{-1,1\}$ The joint distribution of (X,Y) is denoted $\mathbb{P}$ and the marginal distribution of X is denoted M. Let the observation conditional density $\eta(x) := \Pr(\mathsf{Y}=1|\mathsf{X}=x)$. Thus one can specify an experiment by either $\mathbb{P}$ or (η, M).

If $\eta \in [0,1]$ is the probability of observing the label $y=1$ the *point-wise risk* (or *conditional risk*) of the estimate $v \in \mathcal{V}$ is defined as the η-average of the point-wise loss for v:

$$L(\eta, v) := \mathbb{E}_{\mathsf{Y}\sim\eta}[\ell(\mathsf{Y}, v)] = \eta\ell_1(v) + (1-\eta)\ell_{-1}(v).$$

Here, $\mathsf{Y} \sim \eta$ is a shorthand for labels being drawn from a Bernoulli distribution with parameter η. When $\eta : \mathcal{X} \to [0,1]$ is an observation-conditional density, taking the M-average of the point-wise risk gives the *(full) risk* of the estimator v, now interpreted as a function $v : \mathcal{X} \to \mathcal{V}$:

$$\mathbb{L}(\eta, v, M) := \mathbb{E}_{\mathsf{X}\sim M}[L(\eta(\mathsf{X}), v(\mathsf{X}))].$$

We sometimes write $\mathbb{L}(v, \mathbb{P})$ for $\mathbb{L}(\eta, v, M)$ where (η, M) corresponds to the joint distribution $\mathbb{P}$. We write ℓ, L and $\mathbb{L}$ for the loss, point-wise and full risk throughout this paper. The *Bayes risk* is the minimal achievable value of the risk and is denoted

$$\underline{\mathbb{L}}(\eta, M) := \inf_{v \in \mathcal{V}^{\mathcal{X}}} \mathbb{L}(\eta, v, M) = \mathbb{E}_{\mathsf{X}\sim M}\left[\underline{L}(\eta(\mathsf{X}))\right],$$

where

$$[0,1] \ni \eta \mapsto \underline{L}(\eta) := \inf_{v \in \mathcal{V}} L(\eta, v)$$

is the *point-wise* or *conditional Bayes risk*.

There has been increasing awareness of the importance of the conditional Bayes risk curve $\underline{L}(\eta)$—also known as "generalized entropy" (Grünwald and Dawid, 2004)—in the analysis of losses for probability estimation (Kalnishkan et al., 2004, 2007; Abernethy et al., 2009; Masnadi-Shirazi and Vasconcelos, 2009). Below we will see how it is effectively the curvature of $\underline{L}$ that determines much of the structure of these losses.

3. Losses for Class Probability Estimation

We begin by considering CPE losses, that is, functions $\ell : \{-1,1\} \times [0,1] \to [0,\infty)$ and briefly summarise a number of important existing structural results for *proper losses*—a large, natural class of losses for class probability estimation.

3.1 Proper, Fair, Definite and Regular Losses

There are a few properties of losses for probability estimation that we will require. If $\hat{\eta}$ is to be interpreted as an estimate of the true positive class probability η (i.e., when $y=1$) then it is desirable to require that $L(\eta, \hat{\eta})$ be minimised by $\hat{\eta} = \eta$ for all $\eta \in [0,1]$. Losses that satisfy this constraint are said to be *Fisher consistent* and are known as *proper losses* (Buja et al., 2005; Gneiting and Raftery, 2007). That is, a proper loss ℓ satisfies $\underline{L}(\eta) = L(\eta, \eta)$ for all $\eta \in [0,1]$. A *strictly proper* loss is a proper loss for which the minimiser of $L(\eta, \hat{\eta})$ over $\hat{\eta}$ is unique.

We will say a loss is *fair* whenever

$$\ell_{-1}(0) = \ell_1(1) = 0.$$

That is, there is no loss incurred for perfect prediction. The main place fairness is relied upon is in the integral representation of Theorem 6 where it is used to get rid of some constants of integration. In order to explicitly construct losses from their associated "weight functions" as shown in Theorem 7, we will require that the loss be *definite*, that is, its point-wise Bayes risk for deterministic events (i.e., $\eta = 0$ or $\eta = 1$) must be bounded from below:

$$\underline{L}(0) > -\infty \ , \ \underline{L}(1) > -\infty.$$

Since properness of a loss ensures $\underline{L}(\eta) = L(\eta, \eta)$ we see that a fair proper loss is necessarily definite since $L(0,0) = \ell_{-1}(0) = 0 > -\infty$, and similarly for $L(1,1)$. Conversely, if a proper loss is definite then the finite values $\ell_{-1}(0)$ and $\ell_1(1)$ can be subtracted from $\ell_{-1}(\cdot)$ and $\ell_1(\cdot)$ to make it fair.

Finally, for Theorem 4 to hold at the endpoints of the unit interval, we require a loss to be *regular*;[4] that is,

$$\lim_{\eta \searrow 0} \eta \ell_1(\eta) = \lim_{\eta \nearrow 1} (1-\eta)\ell_{-1}(\eta) = 0.$$

Intuitively, this condition ensures that making mistakes on events that never happen should not incur a penalty. In most of the situations we consider in the remainder of this paper will involve losses which are proper, fair, definite and regular.

3.2 The Structure of Proper Losses

A key result in the study of proper losses is originally due to Shuford et al. (1966) and Staël von Holstein (1970) (confer Aczel and Pfanzagl, 1967) though our presentation follows that of Buja et al. (2005). The following theorem[5] characterises proper losses for probability estimation via a constraint on the relationship between its partial losses.

Theorem 1 (Shuford *et al.*) *Suppose $\ell : \{-1,1\} \times [0,1] \to \mathbb{R}$ is a loss and that its partial losses ℓ_1 and ℓ_{-1} are both differentiable. Then ℓ is a proper loss if and only if for all $\hat{\eta} \in (0,1)$*

$$\frac{-\ell_1'(\hat{\eta})}{1-\hat{\eta}} = \frac{\ell_{-1}'(\hat{\eta})}{\hat{\eta}} = w(\hat{\eta}) \tag{2}$$

for some weight function $w : (0,1) \to \mathbb{R}^+$ *such that $\int_\varepsilon^{1-\varepsilon} w(c)\,dc < \infty$ for all $\varepsilon > 0$.*

The equalities in (2) should be interpreted in the distributional sense.

This simple characterisation of the structure of proper losses has a number of interesting implications. Observe from (2) that if ℓ is proper, given ℓ_1 we can determine ℓ_{-1} or vice versa. Also, the partial derivative of the conditional risk can be seen to be the product of a linear term and the weight function:

Corollary 2 *If ℓ is a differentiable proper loss then for all $\eta \in [0,1]$*

$$\frac{\partial}{\partial \hat{\eta}} L(\eta, \hat{\eta}) = (1-\eta)\ell_{-1}'(\hat{\eta}) + \eta \ell_1'(\hat{\eta}) = (\hat{\eta} - \eta) w(\hat{\eta}). \tag{3}$$

Another corollary, observed by Buja et al. (2005), is that the weight function is related to the curvature of the conditional Bayes risk $\underline{L}$.

4. This is equivalent to the conditions of Savage (1971) and Schervish (1989).
5. This is a restatement of Theorem 1 in Shuford et al. (1966).

Corollary 3 *Let ℓ be a a twice differentiable*[6] *proper loss with weight function w defined as in Equation (2). Then for all $c \in (0,1)$ its conditional Bayes risk $\underline{L}$ satisfies*

$$w(c) = -\underline{L}''(c).$$

One immediate consequence of this corollary is that the conditional Bayes risk for a proper loss is always concave. Along with an extra constraint, this gives another characterisation of proper losses (Savage, 1971; Reid and Williamson, 2009a).

Theorem 4 (Savage) *A loss function ℓ is proper if and only if its point-wise Bayes risk $\underline{L}(\eta)$ is concave and for each $\eta, \hat{\eta} \in (0,1)$*

$$L(\eta, \hat{\eta}) = \underline{L}(\hat{\eta}) + (\eta - \hat{\eta})\underline{L}'(\hat{\eta}).$$

Furthermore if ℓ is regular this characterisation also holds at the endpoints $\eta, \hat{\eta} \in \{0,1\}$.

This link between loss and concave functions makes it easy to establish a connection, as Buja et al. (2005) do, between *regret* $\Delta L(\eta, \hat{\eta}) := L(\eta, \hat{\eta}) - \underline{L}(\eta)$ for proper losses and *Bregman divergences*. The latter are generalisations of distances and are defined in terms of convex functions. Specifically, if $f : \mathcal{S} \to \mathbb{R}$ is a convex function over some convex set $\mathcal{S} \subseteq \mathbb{R}^n$ then its associated Bregman divergence[7] is

$$D_f(s, s_0) := f(s) - f(s_0) - \langle s - s_0, \nabla f(s_0) \rangle$$

for any $s, s_0 \in \mathcal{S}$, where $\nabla f(s_0)$ is the gradient of f at s_0. By noting that over $\mathcal{S} = [0,1]$ we have $\nabla f = f'$, these definitions lead immediately to the following corollary of Theorem 4.

Corollary 5 *If ℓ is a proper loss then its regret is the Bregman divergence associated with $f = -\underline{L}$. That is,*

$$\Delta L(\eta, \hat{\eta}) = D_{-\underline{L}}(\eta, \hat{\eta}).$$

Many of the above results can be observed graphically by plotting the conditional risk for a proper loss as in Figure 1. Here we see the two partial losses on the left and right sides of the figure are related, for each fixed $\hat{\eta}$, by the linear map $\eta \mapsto L(\eta, \hat{\eta}) = (1-\eta)\ell_{-1}(\hat{\eta}) + \eta\ell_1(\hat{\eta})$. For each fixed η the properness of ℓ requires that these convex combinations of the partial losses (each slice parallel to the left and right faces) are minimised when $\hat{\eta} = \eta$. Thus, the lines joining the partial losses are tangent to the conditional Bayes risk curve $\eta \mapsto \underline{L}(\eta) = L(\eta, \eta)$ shown above the dotted diagonal. Since the conditional Bayes risk curve is the lower envelope of these tangents it is necessarily concave. The coupling of the partial losses via the tangents to the conditional Bayes risk curve demonstrates why much of the structure of proper losses is determined by the curvature of $\underline{L}$—that is, by the weight function w.

The relationship between a proper loss and its associated weight function is captured succinctly by Schervish (1989) via the following representation of proper losses as a weighted integral of the cost-weighted misclassification losses ℓ_c defined in (1). The reader is referred to Reid and Williamson (2009b) for the details, proof and the history of this result.

6. The restriction to differentiable losses can be removed in most cases if generalised weight functions—that is, possibly infinite but defining a measure on $(0,1)$—are permitted. For example, the weight function for the 0-1 loss is $w(c) = \delta(c - \frac{1}{2})$.

7. A concise summary of Bregman divergences and their properties is given by Banerjee et al. (2005, Appendix A).

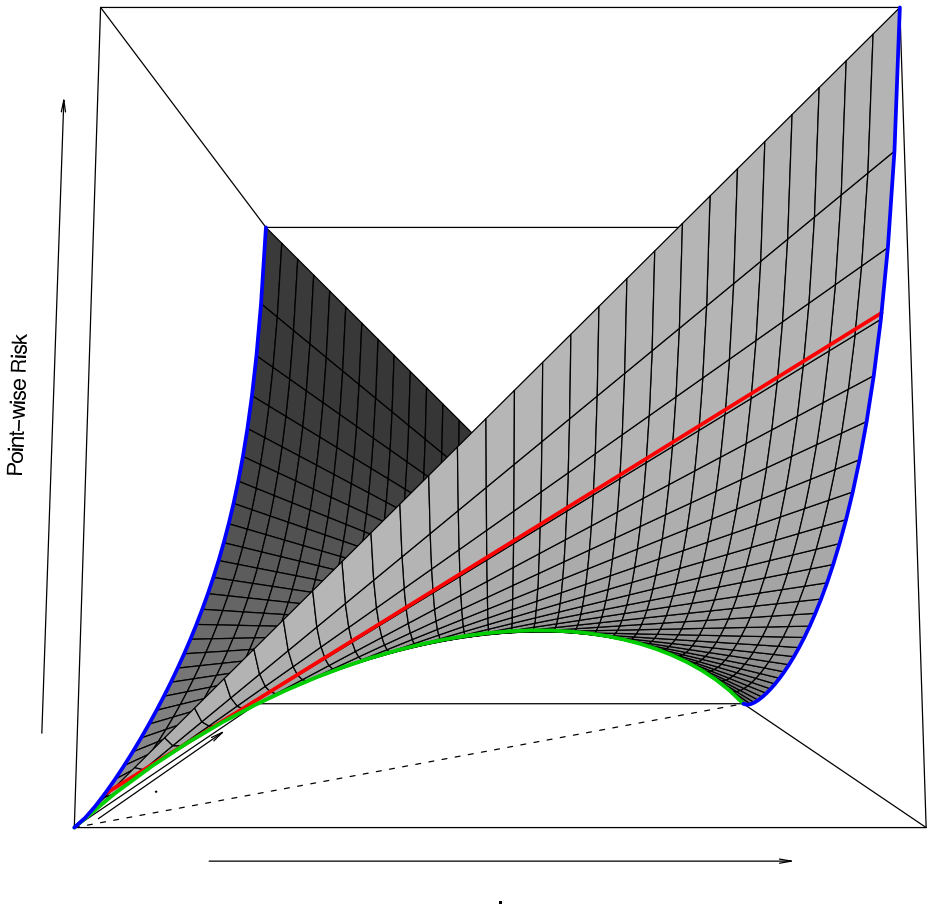


Figure 1: The structure of the conditional risk $L(\eta,\hat{\eta})$ for a proper loss (surface). The loss is log loss and its partials $\ell_{-1}(\hat{\eta}) = -\log(\hat{\eta})$ and $\ell_1(\hat{\eta}) = -\log(1-\hat{\eta})$ shown on the left and right faces of the box. The conditional Bayes risk is the curve on the surface above the dotted line $\hat{\eta} = \eta$. The line connecting points on the partial loss curves shows the conditional risk for a fixed prediction $\hat{\eta}$.

Theorem 6 (Schervish) *Let $\ell : \mathcal{Y} \times [0,1] \to \mathbb{R}$ be a fair, proper loss. Then for each $\hat{\eta} \in (0,1)$ and $y \in \mathcal{Y}$*

$$\ell(y,\hat{\eta}) = \int_0^1 \ell_c(y,\hat{\eta})\, w(c)\, dc, \tag{4}$$

where $w = -\underline{L}''$. Conversely, if ℓ is defined by (4) for some weight function $w : (0,1) \to [0,\infty)$ then it is proper.

Some example losses and their associated weight functions are given in Table 1. Buja et al. (2005) show that ℓ is strictly proper if and only if $w(c) > 0$ in the sense that w has non-zero mass on every open subset of $(0,1)$. The following theorem from Reid and Williamson (2009a) shows how to explicitly construct a loss in terms of a weight function.

$w(c)$	$\ell_{-1}(\hat{\eta})$	$\ell_1(\hat{\eta})$	Loss
$2\delta\left(\frac{1}{2}-c\right)$	$[\![\hat{\eta} > \frac{1}{2}]\!]$	$[\![\hat{\eta} \leq \frac{1}{2}]\!]$	0-1
$\delta(c-c_0)$	$c_0[\![\hat{\eta} \geq c_0]\!]$	$(1-c_0)[\![\hat{\eta} < c_0]\!]$	$\ell_{c_0}, c_0 \in [0,1]$
$\frac{1}{(1-c)^2 c}$	$\left[2\ln(1-\hat{\eta}) + \frac{\hat{\eta}}{1-\hat{\eta}}\right]$	$\left[\ln\frac{1-\hat{\eta}}{\hat{\eta}} - 1\right]$	—
1	$\hat{\eta}^2/2$	$(1-\hat{\eta})^2/2$	Square
$\frac{1}{(1-c)c}$	$-\ln(1-\hat{\eta})$	$-\ln(\hat{\eta})$	Log
$\frac{1}{(1-c)^2 c^2}$	$\left[\ln((1-\hat{\eta})\hat{\eta}) - \frac{1-2\hat{\eta}}{\hat{\eta}}\right]$	$\left[\ln((1-\hat{\eta})\hat{\eta}) + \frac{1-2\hat{\eta}}{\hat{\eta}}\right]$	—
$\frac{1}{[(1-c)c]^{3/2}}$	$2\sqrt{\frac{\hat{\eta}}{1-\hat{\eta}}}$	$2\sqrt{\frac{1-\hat{\eta}}{\hat{\eta}}}$	Boosting

Table 1: Weight functions and associated partial losses.

Theorem 7 (Reid and Williamson) *Given a weight function $w : [0,1] \to [0,\infty)$, let $W(t) = \int^t w(c)\,dc$ and $\overline{W}(t) = \int^t W(c)\,dc$. Then the loss ℓ_w defined by*

$$\ell_w(y,\hat{\eta}) = -\overline{W}(\hat{\eta}) - (y-\hat{\eta})W(\hat{\eta})$$

is a proper loss. Additionally, if $\overline{W}(0)$ and $\overline{W}(1)$ are both finite then

$$\ell_w(y,\hat{\eta}) + (\overline{W}(1) - \overline{W}(0))y + \overline{W}(0)$$

is a fair, proper loss.

Observe that if w and v are weight functions which differ on a set of measure zero then they will lead to the same loss. A simple corollary to Theorem 6 is that the partial losses are given by

$$\ell_1(\hat{\eta}) = \int_{\hat{\eta}}^1 (1-c)w(c)dc \;\text{ and }\; \ell_{-1}(\hat{\eta}) = \int_0^{\hat{\eta}} cw(c)dc. \tag{5}$$

A similar[8] integral representation of the partial losses can also be found in Shuford et al. (1966, Theorem 2) and Staël von Holstein (1970).

3.3 Symmetric Losses

We will say a loss is *symmetric* if $\ell_1(\hat{\eta}) = \ell_{-1}(1-\hat{\eta})$ for all $\hat{\eta} \in [0,1]$. We say a weight function for a proper loss or the conditional Bayes risk is *symmetric* if $w(c) = w(1-c)$ or $\underline{L}(c) = \underline{L}(1-c)$ for all $c \in [0,1]$. Perhaps unsurprisingly, an immediate consequence of Theorem 1 is that these two notions are identical.[9]

8. The weight function h in Theorem 2 of Shuford et al. (1966) is related to the w here by $h(c) = (1-c)w(c)$.
9. The relationship between a symmetric $\underline{L}$ and symmetric behaviour of the loss has been previously recognised by Masnadi-Shirazi and Vasconcelos (2009).

Corollary 8 *A proper loss is symmetric if and only if its weight function is symmetric.*

Proof If ℓ is symmetric, then $\ell_1'(\hat{\eta}) = -\ell_{-1}'(1-\hat{\eta})$ and so Equation (2) implies $w(1-\hat{\eta}) = \frac{\ell_{-1}'(1-\hat{\eta})}{1-\hat{\eta}} = \frac{-\ell_1'(\hat{\eta})}{1-\hat{\eta}} = w(\hat{\eta})$. Conversely, the symmetry of w applied to Equation (5) establishes the symmetry of ℓ. ■

Requiring a loss to be proper and symmetric constrains the partial losses significantly. Properness alone completely specifies one partial loss from the other. Now suppose in addition that ℓ is symmetric. Combining $\ell_1(\hat{\eta}) = \ell_{-1}(1-\hat{\eta})$ with (2) implies

$$\ell_{-1}'(1-\hat{\eta}) = \frac{1-\hat{\eta}}{\hat{\eta}}\ell_{-1}'(\hat{\eta}). \tag{6}$$

This shows that ℓ_{-1} is completely determined by $\ell_{-1}(\hat{\eta})$ for $\hat{\eta} \in [0, \frac{1}{2}]$ (or $\hat{\eta} \in [\frac{1}{2}, 1]$). Thus in order to specify a symmetric proper loss, one needs to only specify one of the partial losses on one half of the interval $[0,1]$. Assuming ℓ_{-1} is continuous at $\frac{1}{2}$ (or equivalently that w has no atoms at $\frac{1}{2}$), by integrating both sides of (6) we can derive an explicit formula for the other half of ℓ_{-1} in terms of that which is specified:

$$\ell_{-1}(\hat{\eta}) = \ell_{-1}(\tfrac{1}{2}) + \int_{\frac{1}{2}}^{\hat{\eta}} \frac{x}{1-x}\ell_{-1}'(1-x)dx, \tag{7}$$

which works for determining ℓ_{-1} on either $[0, \frac{1}{2}]$ or $[\frac{1}{2}, 1]$ when ℓ_{-1} is specified on $[\frac{1}{2}, 1]$ or $[0, \frac{1}{2}]$ respectively (recalling the usual convention that $\int_a^b = -\int_b^a$). We have thus shown:

Theorem 9 *If a loss is proper and symmetric, then it is completely determined by specifying one of the partial losses on half the unit interval (either* $[0, \frac{1}{2}]$ *or* $[\frac{1}{2}, 0]$*) and using (6) and (7).*

We demonstrate (7) with four examples. Suppose that $\ell_{-1}(\hat{\eta}) = \frac{1}{1-\hat{\eta}}$ for $\hat{\eta} \in [0, \frac{1}{2}]$. Then one can readily determine the complete partial loss to be

$$\ell_{-1}(\hat{\eta}) = \frac{[\![\hat{\eta} \leq \frac{1}{2}]\!]}{1-\hat{\eta}} + [\![\hat{\eta} > \tfrac{1}{2}]\!]\left(2 + \log\frac{\hat{\eta}}{1-\hat{\eta}}\right).$$

Suppose instead that $\ell_{-1}(\hat{\eta}) = \frac{1}{1-\hat{\eta}}$ for $\hat{\eta} \in [\frac{1}{2}, 1]$. In that case we obtain

$$\ell_{-1}(\hat{\eta}) = [\![\hat{\eta} \leq \frac{1}{2}]\!]\left(2 + \log\frac{\hat{\eta}}{1-\hat{\eta}}\right) + \frac{[\![\hat{\eta} \geq \frac{1}{2}]\!]}{1-\hat{\eta}}.$$

Suppose $\ell_{-1}(\hat{\eta}) = \frac{1}{(1-\hat{\eta})^2}$ for $\hat{\eta} \in [0, \frac{1}{2}]$. Then one can determine that

$$\ell_{-1}(\hat{\eta}) = \frac{[\![\hat{\eta} < \frac{1}{2}]\!]}{(1-\hat{\eta})^2} + \frac{[\![\hat{\eta} \geq \frac{1}{2}]\!]\,(4 + 2(2\hat{\eta} + \hat{\eta}\log\hat{\eta} - \hat{\eta}\log(1-\hat{\eta}) - 1))}{\hat{\eta}}.$$

Finally consider specifying that $\ell_{-1}(\hat{\eta}) = \hat{\eta}$ for $\hat{\eta} \in [0, \frac{1}{2}]$. In this case we obtain that

$$\ell_{-1}(\hat{\eta}) = [\![\hat{\eta} \leq \tfrac{1}{2}]\!]\hat{\eta} + [\![\hat{\eta} \geq \tfrac{1}{2}]\!]\,(1 - \log 2 - \hat{\eta} - \log(1-\hat{\eta})).$$

4. Composite Losses

General loss functions are often constructed with the aid of a *link function*. For a particular set of prediction values $\mathcal{V}$ this is any continuous mapping $\psi\colon [0,1] \to \mathcal{V}$. In this paper, our focus will be *composite losses* for binary class probability estimation. These are the composition of a CPE loss $\ell\colon \{-1,1\} \times [0,1] \to \mathbb{R}$ and the inverse of a *link function* ψ, an invertible mapping from the unit interval to some range of values. Unless stated otherwise we will assume $\psi\colon [0,1] \to \mathbb{R}$. We will denote a composite loss by

$$\ell^{\psi}(y,v) := \ell(y,\psi^{-1}(v)). \tag{8}$$

The classical motivation for link functions (McCullagh and Nelder, 1989) is that often in estimating η one uses a parametric representation of $\hat{\eta}\colon \mathcal{X} \to$[0,1] which has a natural scale not matching $[0,1]$. Traditionally one writes $\hat{\eta} = \psi^{-1}(\hat{h})$ where ψ^{-1} is the "inverse link" (and ψ is of course the forward link). The function $\hat{h}\colon \mathcal{X} \to \mathbb{R}$ is the *hypothesis*. Often $\hat{h} = \hat{h}_{\boldsymbol{\alpha}}$ is parametrised linearly in a parameter vector $\boldsymbol{\alpha}$. In such a situation it is computationally convenient if $\ell(y,\psi^{-1}(\hat{h}))$ is convex in $\hat{h}$ (which implies it is convex in $\boldsymbol{\alpha}$ when $\hat{h}_{\boldsymbol{\alpha}}$ is linear in $\boldsymbol{\alpha}$). The idea of a link function is not as well known as it should be and is thus reinvented—see for example Granger and Machina (2006).

Often one will choose the loss first (tailoring its properties by the weighting given according to $w(c)$), and *then* choose the link somewhat arbitrarily to map the hypotheses appropriately. An interesting alternative perspective arises in the literature on "elicitability". Lambert et al. (2008)[10] provide a general characterisation of proper scoring rules (i.e., losses) for general *properties* of distributions, that is, continuous and locally non-constant functions Γ which assign a real value to each distribution over a finite sample space. In the binary case, these properties provide another interpretation of links that is complementary to the usual one that treats the inverse link ψ^{-1} as a way of interpreting scores as class probabilities.

To see this, we first identify distributions over $\{-1,1\}$ with the probability η of observing 1. In this case properties are continuous, locally non-constant maps $\Gamma : [0,1] \to \mathbb{R}$. When a link function ψ is continuous it can therefore be interpreted as a property since its assumed invertibility implies it is locally non-constant. A property Γ is said to be *elicitable* whenever there exists a strictly proper loss ℓ for it so that the composite loss ℓ^{Γ} satisfies for all $\hat{\eta} \neq \eta$

$$L^{\Gamma}(\eta,\hat{\eta}) := \mathbb{E}_{Y\sim\eta}[\ell^{\Gamma}(Y,\hat{\eta})] > L^{\Gamma}(\eta,\eta).$$

Theorem 1 of Lambert et al. (2008) shows that Γ is elicitable if and only if $\Gamma^{-1}(r)$ is convex for all $r \in \text{range}(\Gamma)$. This immediately gives us a characterisation of "proper" link functions: those that are both continuous and have convex level sets in $[0,1]$—they are the non-decreasing continuous functions. Thus in Lambert's perspective, one chooses a "property" first (i.e., the invertible link) and *then* chooses the proper loss.

4.1 Proper Composite Losses

We will call a composite loss ℓ^{ψ} (8) a *proper composite loss* if ℓ in (8) is a proper loss for class probability estimation. As in the case for losses for probability estimation, the requirement that a composite loss be proper imposes some constraints on its partial losses. Many of the results for proper losses carry over to composite losses with some extra factors to account for the link function.

10. See also Gneiting (2009).

Theorem 10 *Let $\lambda = \ell^\psi$ be a composite loss with differentiable and strictly monotone link ψ and suppose the partial losses $\lambda_{-1}(v)$ and $\lambda_1(v)$ are both differentiable. Then λ is a proper composite loss if and only if there exists a weight function $w : (0,1) \to \mathbb{R}^+$ such that for all $\hat{\eta} \in (0,1)$*

$$\frac{-\lambda_1'(\psi(\hat{\eta}))}{1-\hat{\eta}} = \frac{\lambda_{-1}'(\psi(\hat{\eta}))}{\hat{\eta}} = \frac{w(\hat{\eta})}{\psi'(\hat{\eta})} =: \rho(\hat{\eta}), \tag{9}$$

where equality is interpreted in the distributional sense. Furthermore, $\rho(\hat{\eta}) \geq 0$ for all $\hat{\eta} \in (0,1)$.

Proof This is a direct consequence of Theorem 1 for proper losses for probability estimation and the chain rule applied to $\ell_y(\hat{\eta}) = \lambda_y(\psi(\hat{\eta}))$. Since ψ is assumed to be strictly monotonic we know $\psi' > 0$ and so, since $w \geq 0$ we have $\rho \geq 0$. ■

As we shall see, the ratio $\rho(\hat{\eta})$ is a key quantity in the analysis of proper composite losses. For example, Corollary 2 has natural analogue in terms of ρ that will be of use later. It is obtained by letting $\hat{\eta} = \psi^{-1}(v)$ and using the chain rule.

Corollary 11 *Suppose ℓ^ψ is a proper composite loss with conditional risk denoted L^ψ. Then*

$$\frac{\partial}{\partial v} L^\psi(\eta, v) = (\psi^{-1}(v) - \eta)\rho(\psi^{-1}(v)). \tag{10}$$

Loosely speaking then, ρ is a "co-ordinate free" weight function for composite losses where the link function ψ is interpreted as a mapping from arbitrary $v \in \mathcal{V}$ to values which can be interpreted as probabilities.

Another immediate corollary of Theorem 10 shows how properness is characterised by a particular relationship between the choice of link function and the choice of partial composite losses.

Corollary 12 *Let $\lambda := \ell^\psi$ be a composite loss with differentiable partial losses λ_1 and λ_{-1}. Then ℓ^ψ is proper if and only if the link ψ satisfies*

$$\psi^{-1}(v) = \frac{\lambda_{-1}'(v)}{\lambda_{-1}'(v) - \lambda_1'(v)}, \quad \forall v \in \mathcal{V}. \tag{11}$$

Proof Substituting $\hat{\eta} = \psi^{-1}(v)$ into (9) yields $-\psi^{-1}(v)\lambda_1'(v) = (1-\psi^{-1}(v))\lambda_{-1}'(v)$ and solving this for $\psi^{-1}(v)$ gives the result. ■

These results give some insight into the "degrees of freedom" available when specifying proper composite losses. Theorem 10 shows that the partial losses are completely determined once the weight function w and ψ (up to an additive constant) is fixed. Corollary 12 shows that for a given link ψ one can specify one of the partial losses λ_y but then properness fixes the other partial loss λ_{-y}. Similarly, given an arbitrary choice of the partial losses, Equation 11 gives the single link which will guarantee the overall loss is proper.

We see then that Corollary 12 provides us with a way of constructing a *reference link* for arbitrary composite losses specified by their partial losses. The reference link can be seen to satisfy

$$\psi(\eta) = \operatorname*{arg\,min}_{v \in \mathbb{R}} L^\psi(\eta, v)$$

for $\eta \in (0,1)$ and thus *calibrates* a given composite loss in the sense of Cohen and Goldszmidt (2004).

Finally, we make a note of an analogue of Corollary 5 for composite losses. It shows that the regret for an arbitrary composite loss is related to a Bregman divergence via its link.

Corollary 13 *Let ℓ^ψ be a proper composite loss with invertible link. Then for all $\eta, \hat{\eta} \in (0,1)$,*

$$\Delta L^\psi(\eta, v) = D_{-\underline{L}}(\eta, \psi^{-1}(v)). \tag{12}$$

This corollary generalises the results due to Zhang (2004b) and Masnadi-Shirazi and Vasconcelos (2009) who considered only margin losses respectively without and with links.

4.2 Derivatives of Composite Losses

We now briefly consider an application of the parametrisation of proper losses as a weight function and link. In order to implement Stochastic Gradient Descent (SGD) algorithms one needs to compute the derivative of the loss with respect to predictions $v \in \mathbb{R}$. Letting $\hat{\eta}(v) = \psi^{-1}(v)$ be the probability estimate associated with the prediction v, we can use (10) when $\eta \in \{0,1\}$ to obtain the update rules for positive and negative examples:

$$\begin{aligned} \frac{\partial}{\partial v}\ell_1^\psi(v) &= (\hat{\eta}(v) - 1)\,\rho(\hat{\eta}(v)), \\ \frac{\partial}{\partial v}\ell_{-1}^\psi(v) &= \hat{\eta}(v)\,\rho(\hat{\eta}(v)). \end{aligned}$$

Given an arbitrary weight function w (which defines a proper loss via Corollary 2 and Theorem 4) and link ψ, the above equations show that one could implement SGD directly parametrised in terms of ρ without needing to explicitly compute the partial losses themselves.

4.3 Margin Losses

The *margin* associated with a real-valued prediction $v \in \mathbb{R}$ and label $y \in \{-1,1\}$ is the product $z = yv$. Any function $\phi\colon \mathbb{R} \to \mathbb{R}^+$ can be used as a *margin loss* by interpreting $\phi(yv)$ as the penalty for predicting v for an instance with label y. Margin losses are inherently symmetric since $yv = (-y)(-v)$ and so the penalty $\phi(yv)$ given for predicting v when the label is y is necessarily the same as the penalty for predicting $-v$ when the label is $-y$. Margin losses have attracted a lot of attention (Bartlett et al., 2000) because of their central role in Support Vector Machines (Cortes and Vapnik, 1995). In this section we explore the relationship between these margin losses and the more general class of composite losses and, in particular, symmetric composite losses.

Recall that a general composite loss is of the form $\ell^\psi(y,v) = \ell(y, \psi^{-1}(v))$ for a loss $\ell\colon \mathcal{Y} \times [0,1] \to [0,\infty)$ and an invertible link $\psi\colon \mathbb{R} \to [0,1]$. We would like to understand when margin losses are suitable for probability estimation tasks. As discussed above, proper losses are a natural class of losses over $[0,1]$ for probability estimation so a natural question in this vein is the following: given a margin loss ϕ can we choose a link ψ so that there exists a proper loss ℓ such that $\phi(yv) = \ell^\psi(y,v)$? In this case the proper loss will be $\ell(y,\hat{\eta}) = \phi(y\psi(\hat{\eta}))$.

The following corollary of Theorem 10 gives necessary and sufficient conditions on the choice of link ψ to guarantee when a margin loss ϕ can be expressed as a proper composite loss.

Corollary 14 *Suppose $\phi\colon \mathbb{R} \to \mathbb{R}$ is a differentiable margin loss. Then, $\phi(yv)$ can be expressed as a proper composite loss $\ell^\psi(y,v)$ if and only if the link ψ satisfies*

$$\psi^{-1}(v) = \frac{\phi'(-v)}{\phi'(-v)+\phi'(v)}.$$

Proof Margin losses, by definition, have partial losses $\lambda_y(v) = \phi(yv)$ which means $\lambda'_1(v) = \phi'(v)$ and $\lambda'_{-1}(v) = -\phi'(-v)$. Substituting these into (11) gives the result. ■

This result provides a way of interpreting predictions v as probabilities $\hat{\eta} = \psi^{-1}(v)$ in a consistent manner, for a problem defined by a margin loss. Conversely, it also guarantees that using any other link to interpret predictions as probabilities will be inconsistent.[11] Another immediate implication is that for a margin loss to be considered a proper loss its link function must be *symmetric* in the sense that

$$\psi^{-1}(-v) = \frac{\phi'(v)}{\phi'(v)+\phi'(-v)} = 1 - \frac{\phi'(-v)}{\phi'(-v)+\phi'(v)} = 1 - \psi^{-1}(v),$$

and so, by letting $v = \psi(\hat{\eta})$, we have $\psi(1-\hat{\eta}) = -\psi(\hat{\eta})$ and thus $\psi(\frac{1}{2}) = 0$.

Corollary 14 can also be seen as a simplified and generalised version of the argument by Masnadi-Shirazi and Vasconcelos (2009) that a concave minimal conditional risk function and a symmetric link completely determines a margin loss.[12]

We now consider a couple of specific margin losses and show how they can be associated with a proper loss through the choice of link given in Corollary 14. The exponential loss $\phi(v) = e^{-v}$ gives rise to a proper loss $\ell(y,\hat{\eta}) = \phi(y\psi(\hat{\eta}))$ via the link

$$\psi^{-1}(v) = \frac{-e^v}{-e^v - e^{-v}} = \frac{1}{1+e^{-2v}}$$

which has non-zero denominator. In this case $\psi(\hat{\eta}) = \frac{1}{2}\log\left(\frac{\hat{\eta}}{1-\hat{\eta}}\right)$ is just the logistic link. Now consider the family of margin losses parametrised by $\alpha \in (0,\infty)$

$$\phi_\alpha(v) = \frac{\log(\exp((1-v)\alpha)+1)}{\alpha}.$$

This family of differentiable convex losses approximates the hinge loss as $\alpha \to \infty$ and was studied in the multiclass case by Zhang et al. (2009). Since these are all differentiable functions with $\phi'_\alpha(v) = \frac{-e^{\alpha(1-v)}}{e^{\alpha(1-v)}+1}$, Corollary 14 and a little algebra gives

$$\psi^{-1}(v) = \left[1 + \frac{e^{2\alpha}+e^{\alpha(1-v)}}{e^{2\alpha}+e^{\alpha(1+v)}}\right]^{-1}.$$

Examining this family of inverse links as $\alpha \to 0$ gives some insight into why the hinge loss is a surrogate for classification but not probability estimation. When $\alpha \approx 0$ an estimate $\hat{\eta} = \psi^{-1}(v) \approx \frac{1}{2}$ for all but very large $v \in \mathbb{R}$. That is, in the limit all probability estimates sit infinitesimally to the right or left of $\frac{1}{2}$ depending on the sign of v.

11. Strictly speaking, if the margin loss has "flat spots"—that is, where $\phi'(v) = 0$—then the choice of link may not be unique.

12. Shen (2005, Section 4.4) seems to have been the first to view margin losses from this more general perspective.

5. Classification Calibration and Proper Losses

The notion of properness of a loss designed for class probability estimation is a natural one. If one is only interested in classification (rather than estimating probabilities) a weaker condition suffices. In this section we will relate the weaker condition to properness.

5.1 Classification Calibration for CPE Losses

We begin by giving a definition of classification calibration for CPE losses (i.e., over the unit interval $[0,1]$) and relate it to composite losses via a link.

Definition 15 *We say a CPE loss ℓ is* classification calibrated at $c \in (0,1)$ *and write ℓ is CC_c if the associated conditional risk L satisfies*

$$\forall \eta \neq c,\ \underline{L}(\eta) < \inf_{\hat{\eta}:\ (\hat{\eta}-c)(\eta-c)\leq 0} L(\eta,\hat{\eta}). \tag{13}$$

The expression constraining the infimum ensures that $\hat{\eta}$ is on the opposite side of c to η, or $\hat{\eta} = c$.

The condition $\mathrm{CC}_{\frac{1}{2}}$ is equivalent to what is called "classification calibrated" by Bartlett et al. (2006) and "Fisher consistent for classification problems" by Lin (2002) although their definitions were only for margin losses. One situation where this more general CC_c notion is more appropriate is when the false positive and false negative costs for a classification problem are unequal.

One might suspect that there is a connection between classification calibrated at c and standard Fisher consistency for class probability estimation losses. The following theorem, which captures the intuition behind the "probing" reduction (Langford and Zadrozny, 2005), characterises the situation.

Theorem 16 *A CPE loss ℓ is CC_c for all $c \in (0,1)$ if and only if ℓ is strictly proper.*

Proof The loss ℓ is CC_c for all $c \in (0,1)$ is equivalent to

$$\begin{aligned}
& \forall c \in (0,1),\ \forall \eta \neq c \begin{cases} \underline{L}(\eta) < \inf_{\hat{\eta}\geq c} L(\eta,\hat{\eta}), & \eta < c \\ \underline{L}(\eta) < \inf_{\hat{\eta}\leq c} L(\eta,\hat{\eta}), & \eta > c \end{cases} \\
\Leftrightarrow\ & \forall \eta \in (0,1),\ \forall c \neq \eta \begin{cases} \forall c > \eta,\ \underline{L}(\eta) < \inf_{\hat{\eta}\geq c} L(\eta,\hat{\eta}) \\ \forall c < \eta,\ \underline{L}(\eta) < \inf_{\hat{\eta}\leq c} L(\eta,\hat{\eta}) \end{cases} \\
\Leftrightarrow\ & \forall \eta \in (0,1),\ \begin{cases} \underline{L}(\eta) < \inf_{\hat{\eta}\geq c>\eta} L(\eta,\hat{\eta}) \\ \underline{L}(\eta) < \inf_{\hat{\eta}\leq c<\eta} L(\eta,\hat{\eta}) \end{cases} \\
\Leftrightarrow\ & \forall \eta \in (0,1),\ \underline{L}(\eta) < \inf_{(\hat{\eta}>\eta) \text{ or } (\hat{\eta}<\eta)} L(\eta,\hat{\eta}) \\
\Leftrightarrow\ & \forall \eta \in (0,1),\ \underline{L}(\eta) < \inf_{\hat{\eta}\neq\eta} L(\eta,\hat{\eta})
\end{aligned}$$

which means L is strictly proper. ■

The following theorem is a generalisation of the characterisation of $\mathrm{CC}_{\frac{1}{2}}$ for margin losses via $\phi'(0)$ due to Bartlett et al. (2006).

Theorem 17 *Suppose ℓ is a loss and suppose that ℓ'_1 and ℓ'_{-1} exist everywhere. Then for any $c \in (0,1)$ ℓ is CC_c if and only if*

$$\ell'_{-1}(c) > 0 \ \text{ and } \ \ell'_1(c) < 0 \ \text{ and } \ c\ell'_1(c) + (1-c)\ell'_{-1}(c) = 0. \tag{14}$$

Proof Since ℓ'_1 and ℓ'_{-1} are assumed to exist everywhere

$$\frac{\partial}{\partial\hat{\eta}}L(\eta,\hat{\eta}) = \eta\ell'_1(\hat{\eta}) + (1-\eta)\ell'_{-1}(\hat{\eta})$$

exists for all $\hat{\eta}$. L is CC_c is equivalent to

$$\begin{aligned}
&\frac{\partial}{\partial\hat{\eta}}L(\eta,\hat{\eta})\bigg|_{\hat{\eta}=c} \begin{cases} >0, & \eta < c < \hat{\eta} \\ <0, & \hat{\eta} < c < \eta \end{cases} \\
\Leftrightarrow\quad &\begin{cases} \forall\eta < c, & \eta\ell'_1(c) + (1-\eta)\ell'_{-1}(c) > 0 \\ \forall\eta > c, & \eta\ell'_1(c) + (1-\eta)\ell'_{-1}(c) < 0 \end{cases} \qquad (15) \\
\Leftrightarrow\quad &\begin{array}{l} c\ell'_1(c) + (1-c)\ell'_{-1}(c) = 0 \\ \text{and } \ell'_{-1}(c) > 0 \text{ and } \ell'_1(c) < 0, \end{array} \qquad (16)
\end{aligned}$$

where we have used the fact that (15) with $\eta = 0$ and $\eta = 1$ respectively substituted implies $\ell'_{-1}(c) > 0$ and $\ell'_1(c) < 0$. ■

If ℓ is proper, then by evaluating (3) at $\eta = 0$ and $\eta = 1$ we obtain $\ell'_1(\hat{\eta}) = -w(\hat{\eta})(1-\hat{\eta})$ and $\ell'_{-1}(\hat{\eta}) = w(\hat{\eta})\hat{\eta}$. Thus (16) implies $-w(c)(1-c) < 0$ and $w(c)c > 0$ which holds if and only if $w(c) \neq 0$. We have thus shown the following corollary.

Corollary 18 *If ℓ is proper with weight w, then for any $c \in (0,1)$,*

$$w(c) \neq 0 \;\Leftrightarrow\; \ell \text{ is } \mathrm{CC}_c.$$

The simple form of the weight function for the cost-sensitive misclassification loss ℓ_{c_0} ($w(c) = \delta(c - c_0)$) gives the following corollary (confer Bartlett et al., 2006):

Corollary 19 *ℓ_{c_0} is CC_c if and only if $c_0 = c$.*

5.2 Calibration for Composite Losses

The translation of the above results to general proper composite losses with invertible differentiable link ψ is straight forward. Condition (13) becomes

$$\forall\eta \neq c,\;\; \underline{L}^{\psi}(\eta) < \inf_{v:\; (\psi^{-1}(v)-c)(\eta-c)\leq 0} L^{\psi}(\eta, \psi^{-1}(v)).$$

Theorem 16 then immediately gives:

Corollary 20 *A composite loss $\ell^{\psi}(\cdot,\cdot) = \ell(\cdot, \psi^{-1}(\cdot))$ with invertible and differentiable link ψ is CC_c for all $c \in (0,1)$ if and only if the associated proper loss ℓ is strictly proper.*

Theorem 17 immediately gives:

Corollary 21 *Suppose ℓ^{ψ} is as in Corollary 20 and that the partial losses ℓ_1 and ℓ_{-1} of the associated proper loss ℓ are differentiable. Then for any $c \in (0,1)$, ℓ^{ψ} is CC_c if and only if (14) holds.*

It can be shown that in the special case of margin losses L_{ϕ}, which satisfy the conditions of Corollary 14 such that they are proper composite losses, Corollary 21 leads to the condition $\phi'(0) < 0$ which is the same as obtained by Bartlett et al. (2006).

6. Convexity of Composite Losses

We have seen that composite losses are defined by the proper loss ℓ and the link ψ. We have further seen from (14) that it is natural to parametrise composite losses in terms of w and ψ', and combine them as ρ. One may wish to choose a weight function w and determine which links ψ lead to a convex loss; or choose a link ψ and determine which weight functions w (and hence proper losses) lead to a convex composite loss. The main result of this section is Theorem 29 answers these questions by characterising the convexity of composite losses in terms of (w,ψ') or ρ.

We first establish some convexity results for losses and their conditional and full risks.

Lemma 22 *Let $\ell : \mathcal{Y} \times \mathcal{V} \to [0,\infty)$ denote an arbitrary loss. Then the following are equivalent:*

1. *$v \mapsto \ell(y,v)$ is convex for all $y \in \{-1,1\}$,*
2. *$v \mapsto L(\eta,v)$ is convex for all $\eta \in [0,1]$,*
3. *$v \mapsto \hat{\mathbb{L}}(v,S) := \frac{1}{|S|}\sum_{(x,y)\in S} \ell(y,v(x))$ is convex for all finite $S \subset \mathcal{X} \times \mathcal{Y}$.*

Proof $1 \Rightarrow 2$: By definition, $L(\eta,v) = (1-\eta)\ell(-1,v) + \eta\ell(1,v)$ which is just a convex combination of convex functions and hence convex.

$2 \Rightarrow 1$: Choose $\eta = 0$ and $\eta = 1$ in the definition of L.

$1 \Rightarrow 3$: For a fixed (x,y), the function $v \mapsto \ell(y,v(x))$ is convex since ℓ is convex. Thus, $\hat{\mathbb{L}}$ is convex as it is a non-negative weighted sum of convex functions.

$3 \Rightarrow 1$: The convexity of $\hat{\mathbb{L}}$ holds for every S so for each $y \in \{-1,1\}$ choose $S = \{(x,y)\}$ for some x. In each case $v \mapsto \hat{\mathbb{L}}(v,S) = \ell(y,v(x))$ is convex as required. ■

The following theorem generalises the corollary on page 12 of Buja et al. (2005) to arbitrary composite losses with invertible links. It has less practical value than the previous lemma since, in general, sums of quasi-convex functions are not necessarily quasi-convex (a function f is quasi-convex if the set $\{x\colon f(x) \geq \alpha\}$ is convex for all $\alpha \in \mathbb{R}$). Thus, assuming properness of the loss ℓ does not guarantee its empirical risk $\hat{\mathbb{L}}(\cdot,S)$ will not have local minima.

Theorem 23 *If $\ell^\psi(y,v) = \ell(y,\psi^{-1}(v))$ is a composite loss where ℓ is proper and ψ is invertible and differentiable then $L^\psi(\eta,v)$ is quasi-convex in v for all $\eta \in [0,1]$.*

Proof Since ℓ is proper we know by Corollary 11 that the conditional Bayes risk satisfies

$$\frac{\partial}{\partial v} L^\psi(\eta,v) \quad = \quad (\psi^{-1}(v) - \eta)\rho(\psi^{-1}(v)).$$

Since ψ is invertible and $\rho \geq 0$ we see that $\frac{\partial}{\partial v} L^\psi(\eta,v)$ only changes sign at $\eta = \psi^{-1}(v)$ and so L^ψ is quasi-convex as required. ■

The following theorem characterises convexity of composite losses with invertible links.

Theorem 24 *Let $\ell^\psi(y,v)$ be a composite loss comprising an invertible link ψ with inverse $q := \psi^{-1}$ and strictly proper loss with weight function w. Assume $q'(\cdot) > 0$. Then $v \mapsto \ell^\psi(y,v)$ is convex for $y \in \{-1,1\}$ if and only if*

$$-\frac{1}{x} \;\leq\; \frac{w'(x)}{w(x)} - \frac{\psi''(x)}{\psi'(x)} \;\leq\; \frac{1}{1-x}, \quad \forall x \in (0,1). \tag{17}$$

This theorem suggests a very natural parametrisation of composite losses is via (w, ψ'). Observe that $w, \psi' \colon [0,1] \to \mathbb{R}^+$. (But also see the comment following Theorem 29.)

Proof We can write the conditional composite loss as

$$L^\psi(\eta, v) = \eta \ell_1(q(v)) + (1-\eta)\ell_{-1}(q(v))$$

and by substituting $q = \psi^{-1}$ into (10) we have

$$\frac{\partial}{\partial v} L^\psi(\eta, v) \quad = \quad w(q(v))q'(v)[q(v) - \eta]. \tag{18}$$

A necessary and sufficient condition for $v \mapsto \ell^\psi(y, v) = L^\psi(y, v)$ to be convex for $y \in \{-1, 1\}$ is that

$$\frac{\partial^2}{\partial v^2} L^\psi(y, v) \geq 0, \quad \forall v \in \mathbb{R},\ \forall y \in \{-1, 1\}.$$

Using (18) the above condition is equivalent to

$$[w(q(v))q'(v)]'(q(v) - [\![y = 1]\!]) + w(q(v))q'(v)q'(v) \ \geq \ 0, \quad \forall v \in \mathbb{R}, \tag{19}$$

where

$$[w(q(v))q'(v)]' := \frac{\partial}{\partial v} w(q(v))q'(v).$$

Inequality (19) is equivalent to (Buja et al., 2005, Equation 39). By further manipulations, we can simplify (19) considerably.

Since $[\![y = 1]\!]$ is either 0 or 1 we equivalently have the two inequalities

$$\begin{aligned} [w(q(v))q'(v)]'q(v) + w(q(v))(q'(v))^2 &\geq 0, \quad \forall v \in \mathbb{R}, \quad (y = -1) \\ [w(q(v))q'(v)]'(q(v) - 1) + w(q(v))(q'(v))^2 &\geq 0, \quad \forall v \in \mathbb{R}, \quad (y = 1), \end{aligned}$$

which we shall rewrite as the pair of inequalities

$$w(q(v))(q'(v))^2 \ \geq \ -q(v)[w(q(v))q'(v)]', \qquad \forall v \in \mathbb{R}, \tag{20}$$

$$w(q(v))(q'(v))^2 \ \geq \ (1 - q(v))[w(q(v))q'(v)]', \quad \forall v \in \mathbb{R}. \tag{21}$$

Observe that if $q(\cdot) = 0$ (resp. $1 - q(\cdot) = 0$) then (20) (resp. (21)) is satisfied anyway because of the assumption on q' and the fact that w is non-negative. It is thus equivalent to restrict consideration to v in the set

$$\{x \colon q(x) \neq 0 \ \text{ and } \ (1 - q(x)) \neq 0\} = q^{-1}((0,1)) = \psi((0,1)).$$

Combining (20) and (21) we obtain the equivalent condition

$$\frac{(q'(v))^2}{1 - q(v)} \ \geq \ \frac{[w(q(v))q'(v)]'}{w(q(v))} \ \geq \ \frac{-(q'(v))^2}{q(v)}, \quad \forall v \in \psi((0,1)), \tag{22}$$

where we have used the fact that $q \colon \mathbb{R} \to [0,1]$ and is thus sign-definite and consequently $-q(\cdot)$ is always negative and division by $q(v)$ and $1 - q(v)$ is permissible since as argued we can neglect the cases when these take on the value zero, and division by $w(q(v))$ is permissible by the assumption of *strict* properness since that implies $w(\cdot) > 0$. Now

$$[w(q(\cdot))q'(\cdot)]' = w'(q(\cdot))q'(\cdot)q'(\cdot) + w(q(\cdot))q''(\cdot)$$

and thus (22) is equivalent to

$$\frac{(q'(v))^2}{1-q(v)} \;\geq\; \frac{w'(q(v))(q'(v))^2 + w(q(v))q''(v)}{w(q(v))} \;\geq\; \frac{-(q'(v))^2}{q(v)}, \quad \forall v \in \psi((0,1)) \tag{23}$$

Now divide all sides of (23) by $(q'(\cdot))^2$ (which is permissible by assumption). This gives the equivalent condition

$$\frac{1}{1-q(v)} \;\geq\; \frac{w'(q(v))}{w(q(v))} + \frac{q''(v)}{(q'(v))^2} \;\geq\; \frac{-1}{q(v)}, \quad \forall v \in \psi((0,1)). \tag{24}$$

Let $x = q(v)$ and so $v = q^{-1}(x) = \psi(x)$. Then (24) is equivalent to

$$\frac{1}{1-x} \;\geq\; \frac{w'(x)}{w(x)} + \frac{q''(\psi(x))}{(q'(\psi(x)))^2} \;\geq\; \frac{-1}{x}, \quad \forall x \in (0,1). \tag{25}$$

Now $\frac{1}{q'(\psi(x))} = \frac{1}{q'(q^{-1}(x))} = (q^{-1})'(x) = \psi'(x)$. Thus (25) is equivalent to

$$\frac{1}{1-x} \;\geq\; \frac{w'(x)}{w(x)} + \Phi_\psi(x) \;\geq\; \frac{-1}{x}, \quad \forall x \in (0,1), \tag{26}$$

where

$$\Phi_\psi(x) := q''\left(\psi(x)\right)\left(\psi'(x)\right)^2.$$

All of the above steps are equivalences. We have thus shown that

$$\text{(26) is true} \;\Leftrightarrow\; v \mapsto L^\psi(y,v) \text{ is convex for } y \in \{-1,1\}$$

where the right hand side is equivalent to the assertion in the theorem by Lemma 22.

Finally we simplify Φ_ψ. We first compute q'' in terms of $\psi = q^{-1}$. Observe that $q' = (\psi^{-1})' = \frac{1}{\psi'(\psi^{-1}(\cdot))}$. Thus

$$\begin{aligned}
q''(\cdot) &= (\psi^{-1})''(\cdot) \\
&= \left(\frac{1}{\psi'(\psi^{-1}(\cdot))}\right)' \\
&= \frac{-1}{\left(\psi'(\psi^{-1}(\cdot))\right)^2}\psi''(\psi^{-1}(\cdot))\left(\psi^{-1}(\cdot)\right)' \\
&= \frac{-1}{\left(\psi'(\psi^{-1}(\cdot))\right)^3}\psi''(\psi^{-1}(\cdot)).
\end{aligned}$$

Thus by substitution

$$\begin{aligned}
\Phi_\psi(\cdot) &= \frac{-1}{\left(\psi'(\psi^{-1}(\psi(\cdot)))\right)^3}\psi''(\psi(\psi^{-1}(\cdot)))\left(\psi'(\cdot)\right)^2 \\
&= \frac{-1}{\left(\psi'(\cdot)\right)^3}\psi''(\cdot)\left(\psi'(\cdot)\right)^2 \\
&= -\frac{\psi''(\cdot)}{\psi'(\cdot)}.
\end{aligned} \tag{27}$$

Substituting the simpler expression (27) for Φ_ψ into (26) completes the proof. ■

Lemma 25 *If q is affine then $\Phi_\psi = 0$.*

Proof Using (27), this is immediate since in this case $\psi''(\cdot) = 0$. ■

Corollary 26 *Composite losses with a linear link (including as a special case the identity link) are convex if and only if*

$$-\frac{1}{x} \leq \frac{w'(x)}{w(x)} \leq \frac{1}{1-x}, \quad \forall x \in (0,1).$$

6.1 Canonical Links

Buja et al. (2005) introduced the notion of a *canonical link* defined by $\psi'(v) = w(v)$. The canonical link corresponds to the notion of "matching loss" as developed by Helmbold et al. (1999) and Kivinen and Warmuth (2001). Note that choice of canonical link implies $\rho(c) = w(c)/\psi'(c) = 1$.

Lemma 27 *Suppose ℓ is a proper loss with weight function w and ψ is the corresponding canonical link, then*

$$\Phi_\psi(x) = -\frac{w'(x)}{w(x)}. \tag{28}$$

Proof Substitute $\psi' = w$ into (27). ■

This lemma gives an immediate proof of the following result due to Buja et al. (2005).

Theorem 28 *A composite loss comprising a proper loss with weight function w combined with its canonical link is always convex.*

Proof Substitute (28) into (17) to obtain

$$-\frac{1}{x} \leq 0 \leq \frac{1}{1-x}, \quad \forall x \in (0,1)$$

which holds for any w. ■

An alternative view of canonical links is given in Appendix B.

6.2 A Simpler Characterisation of Convex Composite Losses

The following theorem prrovides a simpler characterisation of the convexity of composite losses. Noting that loss functions can be multiplied by a scalar without affecting what a learning algorithm will do, it is convenient to normalise them. If w satisfies (17) then so does αw for all $\alpha \in (0,\infty)$. Thus without loss of generality we will normalise w such that $w(\frac{1}{2}) = 1$. We chose to normalise about $\frac{1}{2}$ for two reasons: symmetry and the fact that w can have non-integrable singularities at 0 and 1; see, for example, Buja et al. (2005).

Theorem 29 *Consider a proper composite loss ℓ^ψ with invertible link ψ and (strictly proper) weight w normalised such that $w(\frac{1}{2}) = 1$. Then ℓ is convex if and only if*

$$\frac{\psi'(x)}{x} \lesseqgtr 2\psi'(\tfrac{1}{2})\, w(x) \lesseqgtr \frac{\psi'(x)}{1-x}, \quad \forall x \in (0,1), \tag{29}$$

where $\lesseqgtr$ denotes $\leq$ for $x \geq \frac{1}{2}$ and denotes $\geq$ for $x \leq \frac{1}{2}$.

Observe that the condition (29) is equivalent to

$$\frac{1}{2\psi'(\frac{1}{2})x} \begin{matrix}\leqq\\ \geqq\end{matrix} \rho(x) \begin{matrix}\leqq\\ \geqq\end{matrix} \frac{1}{2\psi'(\frac{1}{2})(1-x)}, \quad \forall x \in (0,1),$$

which suggests the importance of the function $\rho(\cdot)$.

Proof Observing that $\frac{w'(x)}{w(x)} = (\log w)'(x)$ we let $g(x) := \log w(x)$. Observe that $g(v) = \int_{\frac{1}{2}}^{v} g'(x)dx + g(\frac{1}{2})$ and $g(\frac{1}{2}) = \log w(\frac{1}{2}) = 0$. From Theorem 24, we know that ℓ is convex iff (17) holds. Using the newly introduced notation, this is equivalent to

$$-\frac{1}{x} - \Phi_\psi(x) \;\leq\; g'(x) \;\leq\; \frac{1}{1-x} - \Phi_\psi(x).$$

For $v \geq \frac{1}{2}$ we thus have

$$\int_{\frac{1}{2}}^{v} -\frac{1}{x} - \Phi_\psi(x)dx \;\leq\; g(v) \;\leq\; \int_{\frac{1}{2}}^{v} \frac{1}{1-x} - \Phi_\psi(x)dx.$$

Similarly, for $v \leq \frac{1}{2}$ we have

$$\int_{\frac{1}{2}}^{v} -\frac{1}{x} - \Phi_\psi(x)dx \;\geq\; g(v) \;\geq\; \int_{\frac{1}{2}}^{v} \frac{1}{1-x} - \Phi_\psi(x)dx,$$

and thus

$$-\ln v - \ln 2 - \int_{\frac{1}{2}}^{v} \Phi_\psi(x)dx \begin{matrix}\leqq\\ \geqq\end{matrix} g(v) \begin{matrix}\leqq\\ \geqq\end{matrix} -\ln 2 - \ln(1-v) - \int_{\frac{1}{2}}^{v} \Phi_\psi(x)dx.$$

Since $\exp(\cdot)$ is monotone increasing we can apply it to all terms and obtain

$$\frac{1}{2v}\exp\left(-\int_{\frac{1}{2}}^{v} \Phi_\psi(x)dx\right) \begin{matrix}\leqq\\ \geqq\end{matrix} w(v) \begin{matrix}\leqq\\ \geqq\end{matrix} \frac{1}{2(1-v)}\exp\left(-\int_{\frac{1}{2}}^{v} \Phi_\psi(x)dx\right). \tag{30}$$

Now

$$\int_{\frac{1}{2}}^{v} \Phi_\psi(x)dv = \int_{\frac{1}{2}}^{v} -\frac{\psi''(x)}{\psi'(x)}dx = -\int_{\frac{1}{2}}^{v} (\log\psi')'(x)dx = -\log\psi'(v) + \log\psi'(\tfrac{1}{2})$$

and so

$$\exp\left(-\int_{\frac{1}{2}}^{v} \Phi_\psi(x)dx\right) = \frac{\psi'(v)}{\psi'(\frac{1}{2})}.$$

Substituting into (30) completes the proof. ■

If ψ is the identity (i.e., if ℓ^ψ is itself proper) we get the simpler constraints

$$\frac{1}{2x} \begin{matrix}\leqq\\ \geqq\end{matrix} w(x) \begin{matrix}\leqq\\ \geqq\end{matrix} \frac{1}{2(1-x)}, \quad \forall x \in (0,1), \tag{31}$$

which are illustrated as the shaded region in Figure 2. Observe that the (normalised) weight function for squared loss is $w(c) = 1$ which is indeed within the shaded region as one would expect.

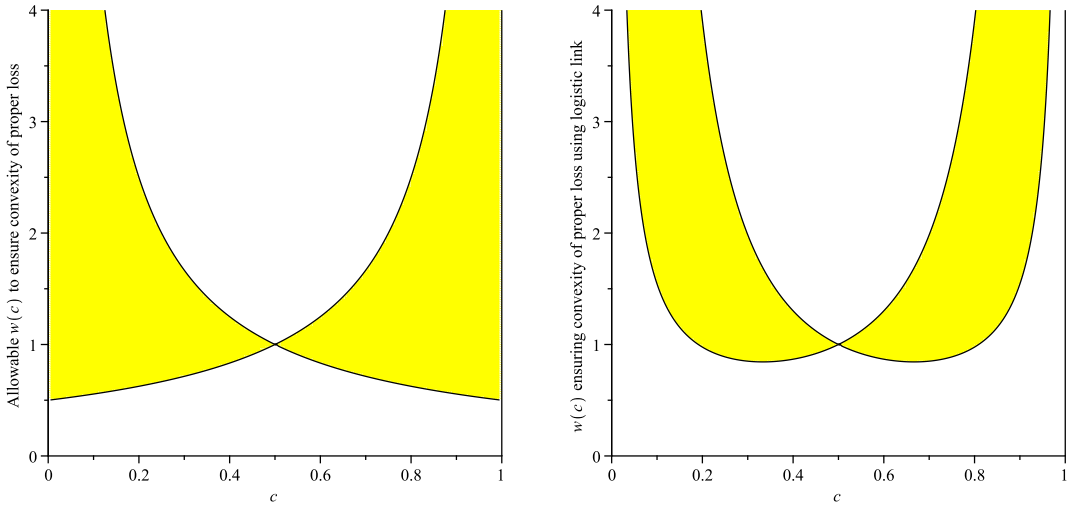


Figure 2: Allowable normalised weight functions to ensure convexity of composite loss functions with identity link (left) and logistic link (right).

Consider the link $\psi^{\text{logit}}(c) := \log\left(\frac{c}{1-c}\right)$ with corresponding inverse link $q(c) = \frac{1}{1+e^{-c}}$. One can check that $\psi'(c) = \frac{1}{c(1-c)}$. Thus the constraints on the weight function w to ensure convexity of the composite loss are

$$\frac{1}{8x^2(1-x)} \leqq w(x) \geqq \frac{1}{8x(1-x)^2}, \quad \forall x \in (0,1).$$

This is shown graphically in Figure 2. One can compute similar regions for any link. Two other examples are the Complementary Log-Log link $\psi^{\text{CLL}}(x) = \log(-\log(1-x))$ (confer McCullagh and Nelder, 1989), the "square link" $\psi^{\text{sq}}(x) = x^2$ and the "cosine link" $\psi^{\cos}(x) = 1 - \cos(\pi x)$. All of these are illustrated in Figure 3. The reason for considering these last two rather unusual links is to illustrate the following fact. Observing that the allowable region in Figure 2 precludes weight functions that approach zero at the endpoints of the interval, and noting that in order to well approximate the behaviour of 0-1 loss (with its weight function being $w_{0-1}(c) = \delta(c - \frac{1}{2})$) one would like a weight function that does indeed approach zero at the end points, it is natural to ask what constraints are imposed upon a link ψ such that a composite loss with that link and a weight function $w(c)$ such that

$$\lim_{c \searrow 0} w(c) = \lim_{c \nearrow 1} w(c) = 0 \tag{32}$$

is convex. Inspection of (29) reveals it is necessary that $\psi'(x) \to 0$ as $x \to 0$ and $x \to 1$. Such ψ necessarily have bounded range and thus the inverse link ψ^{-1} is only defined on a finite interval and furthermore the gradient of ψ^{-1} will be arbitrarily large. If one wants inverse links defined on the whole real line (such as the logistic link) then one can not obtain a convex composite link with the associated proper loss having a weight function satisfying (32). Thus one can not choose an effectively usable link to ensure convexity of a proper loss that is arbitrarily "close to" 0-1 loss in the sense of the corresponding weight functions.

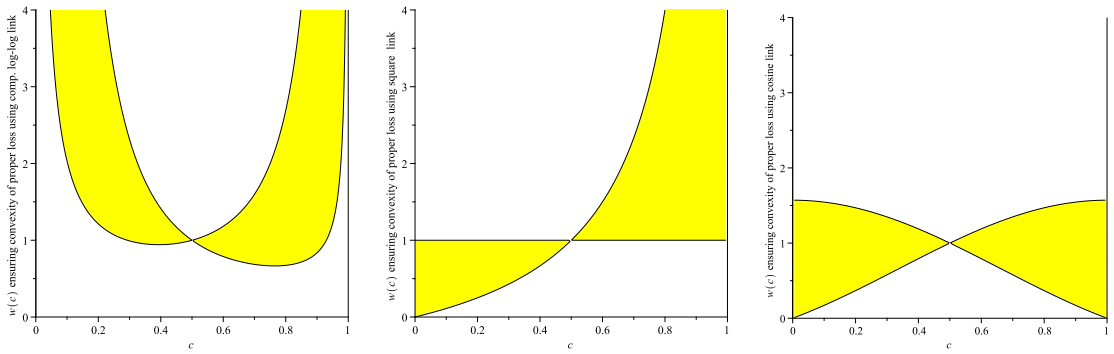


Figure 3: Allowable normalised weight functions to ensure convexity of loss functions with complementary log-log, square and cosine links.

Corollary 30 *If a loss is proper and convex, then it is strictly proper.*

The proof of Corollary 30 makes use of the following special case of the Gronwall style Lemma 1.1.1 of Bainov and Simeonov (1992).

Lemma 31 *Let $b\colon \mathbb{R} \to \mathbb{R}$ be continuous for $t \geq \alpha$. Let $v(t)$ be differentiable for $t \geq \alpha$ and suppose $v'(t) \leq b(t)v(t)$, for $t \geq \alpha$ and $v(\alpha) \leq v_0$. Then for $t \geq \alpha$,*

$$v(t) \leq v_0 \exp\left(\int_\alpha^t b(s)ds\right).$$

Proof (Corollary 30) Observe that the RHS of (17) implies

$$w'(v) \leq \frac{w(v)}{1-v},\ v \geq 0.$$

Suppose $w(0) = 0$. Then $v_0 = 0$ and the setting $\alpha = 0$ the lemma implies

$$w(t) \leq v_0 \exp\left(\int_0^t \frac{1}{1-s}ds\right) = \frac{v_0}{1-t} = 0, \quad t \in (0,1].$$

Thus if $w(0) = 0$ then $w(t) = 0$ for all $t \in (0,1)$. Choosing any other $\alpha \in (0,1)$ leads to a similar conclusion. Thus if $w(t) = 0$ for some $t \in [0,1)$, $w(s) = 0$ for all $s \in [t,1]$. Hence $w(t) > 0$ for all $t \in [0,1]$ and hence by the remark immediately following Theorem 6 ℓ is strictly proper. ■

6.3 Convexity of Bregman Divergences in their Second Argument

Bregman divergences are always convex in the first argument but only sometimes in their second. Corollary 5 and Equation 31 together characterise when the Bregman divergence $D_\phi(\eta, \hat{\eta})$ defined on $(0,1) \times (0,1)$ is convex in $\hat{\eta}$, providing a more direct result that that in Bauschke and Borwein (2001): Setting $\phi = -\underline{L}$ we immediately obtain that $\hat{\eta} \mapsto D_\phi(\eta, \hat{\eta})$ is convex for all $\eta \in (0,1)$ iff (31) holds, where $w(c) = \phi''(c)$.

7. Choosing a Surrogate Loss

A *surrogate* loss function is a loss function which is not exactly what one wishes to minimise but is easier to work with algorithmically. Convex surrogate losses are often used in place of the 0-1 loss which is not convex.

Surrogate losses have garnered increasing interest in the machine learning community (Zhang, 2004b; Bartlett et al., 2006; Steinwart, 2007; Steinwart and Christmann, 2008). Some of the questions considered to date are bounding the regret of a desired loss in terms of a surrogate ("surrogate regret bounds"—see Reid and Williamson, 2009b and references therein), the relationship between the decision theoretic perspective and the elicitability perspective (Masnadi-Shirazi and Vasconcelos, 2009), and efficient algorithms for minimising convex surrogate margin losses (Nock and Nielsen, 2009b,a).

Typically convex surrogates are used because they lead to convex, and thus tractable, optimisation problems. To date, work on surrogate losses has focussed on margin losses which necessarily are symmetric with respect to false positives and false negatives (Buja et al., 2005). In line with the rest of this paper, our treatment will not be so restricted.

The aim here is put forward some plausible definitions of what it might mean to select a "best" surrogate from a class of losses—for example, the class of proper, convex composite losses. We make use of the weight function perspective and the convexity results given in the previous section to investigate some new definitions for "best" surrogate and put forward some conjectures regarding them.

7.1 The "Best" Surrogate Loss

There are many choices of surrogate loss one can choose. A natural question is thus "which is best?". In order to do this we need to first define how we are evaluating losses as surrogates. To do this we require notation to describe the set of minimisers of the conditional and full risk associated with a loss. Given a loss $\ell\colon \{-1,1\} \times \mathcal{V} \to \mathbb{R}$ its *conditional minimisers at* $\eta \in [0,1]$ is the set

$$H(\ell,\eta) := \{v \in \mathcal{V}\colon L(\eta,v) = \underline{L}(\eta)\}. \tag{33}$$

Given a set of hypotheses $\mathcal{H} \subseteq \mathcal{V}^{\mathcal{X}}$, the (constrained) Bayes optimal risk is

$$\underline{\mathbb{L}}_{\mathcal{H}} := \inf_{h\in\mathcal{H}} \mathbb{L}(h,\mathbb{P}).$$

The *(full) minimisers over* $\mathcal{H}$ *for* $\mathbb{P}$ is the set

$$\mathcal{H}(\ell,\mathbb{P}) := \{h \in \mathcal{H}\colon \mathbb{L}(h) = \underline{\mathbb{L}}_{\mathcal{H}}\},$$

where $\mathcal{H} \subseteq \mathcal{V}^{\mathcal{X}}$ is some restricted set of functions and $\mathbb{L}(h) := \mathbb{E}_{(\mathsf{X},\mathsf{Y})\sim\mathbb{P}}[\ell(\mathsf{Y},h(\mathsf{X}))]$ and the expectation is with respect to $\mathbb{P}$. Given a *reference loss* ℓ_{ref}, we will say the ℓ_{ref}*-surrogate penalty* of a loss ℓ over the function class $\mathcal{H}$ on a problem (η,M) (or equivalently $\mathbb{P}$) is

$$S_{\ell_{\text{ref}}}(\ell,\eta,M) = S_{\ell_{\text{ref}}}(\ell,\mathbb{P}) := \inf_{h\in\mathcal{H}(\ell,\mathbb{P})} \mathbb{L}_{\text{ref}}(h),$$

where it is important to remember that $\mathbb{L}$ is with respect to $\mathbb{P}$. That is, $S_{\ell_{\text{ref}}}(\ell,\mathbb{P})$ is the minimum ℓ_{ref} risk obtainable by a function in $\mathcal{H}$ that minimises the ℓ risk.

Given a fixed experiment $\mathbb{P}$, if $\mathcal{L}$ is a class of losses then the *best surrogate losses in* $\mathcal{L}$ for the reference loss ℓ_{ref} are those that minimise the ℓ_{ref}-surrogate penalty. This definition is motivated by the manner in which surrogate losses are used—one minimizes $\mathbb{L}(h)$ over h to obtain the minimiser h^* and one hopes that $\mathbb{L}_{\text{ref}}(h^*)$ is small. Clearly, if the class of losses contains the reference loss (i.e., $\ell_{\text{ref}} \in \mathcal{L}$) then ℓ_{ref} will be a best surrogate loss. Therefore, the question of best surrogate loss is only interesting when $\ell_{\text{ref}} \notin \mathcal{L}$. One particular case we will consider is when the reference loss is the 0-1 loss and the class of surrogates $\mathcal{L}$ is the set of convex proper losses. Since 0-1 loss is not convex the question of which surrogate is best is non-trivial.

It would be nice if one could reason about the "best" surrogate loss using the conditional perspective (that is working with L instead of $\mathbb{L}$) and in a manner independent of $\mathcal{H}$. It is simple to see why this can not be done. Since all the losses we consider are proper, the minimiser over $\hat{\eta}$ of $L(\eta, \hat{\eta})$ is η. Thus any proper loss would lead to the same $\hat{\eta} \in [0,1]$. It is only the introduction of the restricted class of hypotheses $\mathcal{H}$ that prevents this reasoning being applied for $\mathbb{L}$: restrictions on $h \in \mathcal{H}$ prevent $h(x) = \eta(x)$ for all $x \in \mathcal{X}$. We conclude that the problem of best surrogate loss only makes sense when one both takes expectations over X *and* restricts the class of hypotheses h to be drawn from some set $\mathcal{H} \subsetneq [0,1]^{\mathcal{X}}$.

This reasoning accords with that of Nock and Nielsen (2009b,a) who examined which surrogate to use and proposed a data-dependent scheme that tunes surrogates for a problem. They explicitly considered proper losses and said that "minimizing any [lower-bounded, symmetric proper] loss amounts to the *same* ultimate goal" and concluded that "the crux of the choice of the [loss] relies on data-dependent considerations".

We demonstrate the difficulty of finding a universal best surrogate loss in by constructing a simple example. One can construct experiments (η_1, M) and (η_2, M) and proper losses ℓ_1 and ℓ_2 such that

$$S_{\ell_{0-1}}(\ell_1, (\eta_1, M)) > S_{\ell_{0-1}}(\ell_2, (\eta_1, M)) \text{ but } S_{\ell_{0-1}}(\ell_1, (\eta_2, M)) < S_{\ell_{0-1}}(\ell_2, (\eta_2, M)).$$

(The examples we construct have weight functions that "cross-over" each other; the details are in Appendix A.) However, this does not imply there can not exist a particular convex ℓ^* that minorizes all proper losses in this sense. Indeed, we conjecture that, in the sense described above, there is no best proper, convex surrogate loss.

Conjecture 32 *Given a proper, convex loss ℓ there exists a second proper, convex loss $\ell^* \neq \ell$, a hypothesis class $\mathcal{H}$, and an experiment $\mathbb{P}$ such that $S_{\ell_{0-1}}(\ell^*, \mathbb{P}) < S_{\ell_{0-1}}(\ell, \mathbb{P})$ for the class $\mathcal{H}$.*

To prove the above conjecture it would suffice to show that for a fixed hypothesis class and any pair of losses one can construct two experiments such that one loss minorises the other loss on one experiment and *vice versa* on the other experiment.

Supposing the above conjecture is true, one might then ask for a best surrogate loss for some reference loss ℓ_{ref} in a minimax sense. Formally, we would like the loss $\ell^* \in \mathcal{L}$ such that the worst-case penalty for using ℓ^*,

$$\Upsilon_{\mathcal{L}}(\ell^*) := \sup_{\mathbb{P}} \left\{ S_{\ell_{\text{ref}}}(\ell^*, \mathbb{P}) - \inf_{\ell \in \mathcal{L}} S_{\ell_{\text{ref}}}(\ell, \mathbb{P}) \right\}$$

is minimised. That is, $\Upsilon_{\mathcal{L}}(\ell^*) \leq \Upsilon_{\mathcal{L}}(\ell)$ for all $\ell \in \mathcal{L}$.

7.2 The "Minimal" Symmetric Convex Proper Loss

Theorem 29 suggests an answer to the question "What is the proper convex loss closest to the 0-1 loss?" A way of making this question precise follows. Since ℓ is presumed proper, it has a weight function w. Suppose w.l.o.g. that $w(\frac{1}{2}) = 1$. Suppose the link is the identity. The constraints in (17) imply that the weight function that is most similar to that for 0-1 loss meets the constraints. Thus from (31)

$$w^{\text{minimal}}(c) = \frac{1}{2}\left(\frac{1}{c} \wedge \frac{1}{1-c}\right) \tag{34}$$

is the weight for the convex proper loss closest to 0-1 loss in this sense. It is the weight function that forms the lower envelope of the shaded region in the left diagram of Figure 2. Using (5) one can readily compute the corresponding partial losses explicitly

$$\ell_{-1}^{\text{minimal}}(\hat{\eta}) = \frac{1}{2}\left([\![\hat{\eta} < \tfrac{1}{2}]\!](-\hat{\eta} - \ln(1-\hat{\eta})) + [\![\hat{\eta} \geq \tfrac{1}{2}]\!](\hat{\eta} - 1 - \ln(\tfrac{1}{2}))\right) \tag{35}$$

and

$$\ell_{1}^{\text{minimal}}(\hat{\eta}) = \frac{1}{2}\left([\![\hat{\eta} < \tfrac{1}{2}]\!](-\hat{\eta} - \log(\tfrac{1}{2})) + [\![\hat{\eta} \geq \tfrac{1}{2}]\!](\hat{\eta} - 1 - \ln\hat{\eta})\right). \tag{36}$$

Observe that the partial losses are (in part) linear, which is unsurprising as linear functions are on the boundary of the set convex functions. This loss is also best in another more precise (but ultimately unsatisfactory) sense, as we shall now show.

Surrogate regret bounds are theoretical bounds on the regret of a desired loss (say 0-1 loss) in terms of the regret with respect to a surrogate. Reid and Williamson (2009b) have shown the following (we only quote the simpler symmetric case here):

Theorem 33 *Suppose ℓ is a proper loss with corresponding conditional Bayes risk $\underline{L}$ which is symmetric about $\frac{1}{2}$: $\underline{L}(\frac{1}{2} - c) = \underline{L}(\frac{1}{2} + c)$ for $c \in [0, \frac{1}{2}]$. If the regret for the $\ell_{\frac{1}{2}}$ loss $\Delta L_{\frac{1}{2}}(\eta, \hat{\eta}) = \alpha$, then the regret ΔL with respect to ℓ satisfies*

$$\Delta L(\eta, \hat{\eta}) \geq \underline{L}(\tfrac{1}{2}) - \underline{L}(\tfrac{1}{2} + \alpha). \tag{37}$$

The bound in the theorem can be inverted to upper bound $\Delta L_{\frac{1}{2}}$ given an upper bound on $\Delta L(\eta, \hat{\eta})$. Considering all symmetric proper losses normalised such that $w(\frac{1}{2}) = 1$, the right side of (37) is maximised and thus the bound on $\Delta L_{\frac{1}{2}}$ in terms of ΔL is minimised when $\underline{L}(\frac{1}{2} + \alpha)$ is maximised (over all losses normalised as mentioned). But since $w = -\underline{L}''$, that occurs for the pointwise minimiser of w (subject to $w(\frac{1}{2}) = 1$). Since we are interested in convex losses, the minimising w is given by (34). In this case the right hand side of (37) can be explicitly determined to be $(\frac{\alpha}{2} + \frac{1}{4})\log(2\alpha + 1) - \frac{\alpha}{2}$, and the bound can be inverted to obtain the result that if $\Delta L^{\text{minimal}}(\eta, \hat{\eta}) = x$ then

$$\Delta L_{\frac{1}{2}}(\eta, \hat{\eta}) \leq \frac{1}{2}\exp\left(\text{LambertW}\left(\frac{(4x-1)}{\mathrm{e}}\right) + 1\right) - \frac{1}{2} \tag{38}$$

which is plotted in Figure 4.[13]

The above argument does *not* show that the loss given by (35,36) is the *best* surrogate loss. The reason is that the above is optimising a *bound* on the regret, not the *actual* regret; the argument in

13. The LambertW function is the real-valued solution of $x \mapsto W(x)e^{W(x)}$. It is commonly found in solutions to differential equations, has no closed form. Its details are not relevant to this discussion except for computing Figure 4.

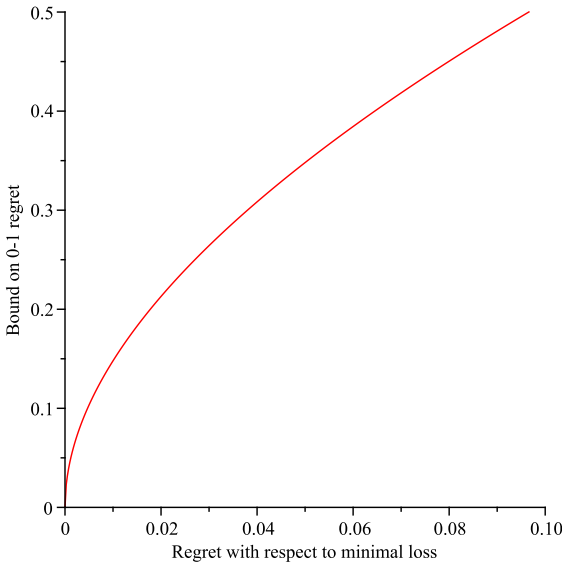


Figure 4: Upper bound on the 0-1 regret in terms of $\Delta L^{\text{minimal}}$ as given by (38).

Appendix A demonstrates there can in general be no universally best surrogate loss (independent of the underlying distribution). Nevertheless it does suggest it is at least worth considering using ℓ^{minimal} as a convex proper surrogate binary loss.

We conjecture that ℓ^{minimal} is somehow special in the class of proper convex losses in some way other than being the pointwise minimiser of weights (and the normalised loss with smallest regret bound with respect to ℓ^{0-1}), but the exact nature of the specialness still eludes us. Perhaps it is optimal in some weaker (minimax) sense. The reason for this suggestion is that it is not hard to show that for reasonable $\mathbb{P}$ there exists $\mathcal{H}$ such that $c \mapsto \mathbb{L}_c(h,\mathbb{P})$ takes on all possible values within the constraints

$$0 \leq \mathbb{L}_c(h,\mathbb{P}) \leq \max(c, 1-c)$$

which follows immediately from the definition of cost-sensitive misclassification loss. Furthermore the example in the appendix below seems to require loss functions whose corresponding weight functions cross over each other and there is no weight function corresponding to a convex proper loss that crosses over w^{minimal}.

8. Conclusions

Composite losses are widely used. As outlined in §1.1, we have characterised a number of aspects of them: their relationship to margin losses, the connection between properness and classification calibration, the constraints symmetry imposes, when composite losses are convex, and natural ways to parametrise them. We have also considered the question of the "best" surrogate loss.

The parametrisation of a composite loss in terms (w,ψ') (or ρ) has advantages over using (ϕ,ψ) or $(\underline{L},\psi)$. As explained by Masnadi-Shirazi and Vasconcelos (2009), the representation in terms of (ϕ,ψ) is in general not unique. The representation in terms of $\underline{L}$ is harder to intuit: whilst indeed the Bayes risk for squared loss and 0-1 loss are "close" (compare the graph of $c \mapsto c(1-c)$ with that of

$c \mapsto c \wedge (1-c)$), by examining their weight functions they are seen to be very different ($w(c) = 1$ versus $w(c) = 2\delta(c - \frac{1}{2})$). We have also seen that on the basis of Theorem 24, the parametrisation (w, ψ') is perhaps the most natural—there is a pleasing symmetry between the loss and the link as they are in this form both parametrised in terms of non-negative weight functions on $[0,1]$. Recall too that the canonical link sets ψ' equal to w.

The observation suggests an alternate inductive principle known as *surrogate tuning*, which seems to have been first suggested by Nock and Nielsen (2009b).[14] The idea of surrogate tuning is simple: noting that the best surrogate depends on the problem, adapt the surrogate you are using to the problem. In order to do so it is important to have a good parametrisation of the loss. The weight function perspective does just that, especially given Theorem 29. It would be straight forward to develop low dimensional parametrisations of w that satisfy the conditions of this theorem which would thus allow a learning algorithm to explore the space of convex losses. One could (taking due care with the subsequent multiple hypothesis testing problem) regularly *evaluate* the 0-1 loss of the hypotheses so obtained. The observations made in Section 4 regarding stochastic gradient descent algorithms may be of help in this regard.

Acknowledgments

This work was motivated in part by a question due to John Langford. Thanks to Fangfang Lu for discussions and finding several bugs in an earlier version. Thanks to Ingo Steinwart for pointing out the η_α trick. Thanks to Tim van Erven and the anonymous reviewers for comments and corrections. This work was supported by the Australian Research Council and NICTA through Backing Australia's Ability.

Appendix A. Example Showing Incommensurability of Two Proper Surrogate Losses

We consider $\mathcal{X} = [0,1]$ with M being uniform on $\mathcal{X}$, and consider the two problems that are induced by

$$\eta_1(x) = x^2 \quad \text{and} \quad \eta_2(x) = \frac{1}{3} + \frac{x}{3}.$$

We use a simple linear hypothesis class

$$\mathcal{H} := \{h_\alpha(x) := \alpha x \colon \alpha \in [0,1]\},$$

with identity link function and consider the two surrogate proper losses ℓ_1 and ℓ_2 with weight functions

$$w_1(c) = \frac{1}{c}, \qquad w_2(c) = \frac{1}{1-c}.$$

These weight functions correspond to the two curves that construct the left diagram in Figure 2. The corresponding conditional losses can be readily calculated to be

$$\begin{aligned} L_1(\eta, h) &:= \eta(h - 1 - \log(h)) + (1-\eta)h \\ L_2(\eta, h) &:= \eta(1-h) + (1-\eta)(-h - \log(1-h)). \end{aligned}$$

14. Surrogate tuning differs from loss *tailoring* (Hand, 1994; Hand and Vinciotti, 2003; Buja et al., 2005) which involves adapting the loss to what you really think is important.

One can numerically compute the parameters for the constrained Bayes optimal for each problem and for each surrogate loss:

$$\begin{aligned}
\alpha^*_{1,1} &= \arg\min_{\alpha\in[0,1]} \mathbb{L}_1(\eta_1, h_\alpha, M) = 0.66666667 \\
\alpha^*_{2,1} &= \arg\min_{\alpha\in[0,1]} \mathbb{L}_2(\eta_1, h_\alpha, M) = 0.81779259 \\
\alpha^*_{1,2} &= \arg\min_{\alpha\in[0,1]} \mathbb{L}_1(\eta_2, h_\alpha, M) = 1.00000000 \\
\alpha^*_{2,1} &= \arg\min_{\alpha\in[0,1]} \mathbb{L}_2(\eta_2, h_\alpha, M) = 0.77763472.
\end{aligned}$$

Furthermore

$$\begin{aligned}
&\mathbb{L}_{0-1}(\eta_1, h_{\alpha^*_{1,1}}, M) = 0.3580272, \qquad \mathbb{L}_{0-1}(\eta_1, h_{\alpha^*_{2,1}}, M) = 0.3033476, \\
&\mathbb{L}_{0-1}(\eta_2, h_{\alpha^*_{1,2}}, M) = 0.4166666, \qquad \mathbb{L}_{0-1}(\eta_2, h_{\alpha^*_{2,2}}, M) = 0.4207872.
\end{aligned}$$

Thus for problem η_1 the surrogate loss L_2 has a constrained Bayes optimal hypothesis $h_{\alpha^*_{2,1}}$ which has a lower 0-1 risk than the constrained Bayes optimal hypothesis $h_{\alpha^*_{1,1}}$ for the surrogate loss L_1. Thus for problem η_1 surrogate L_2 is better than surrogate L_1. However for problem η_2 the situation is reversed: surrogate L_2 is *worse* than surrogate L_1.

Appendix B. An Alternate View of Canonical Links

This appendix contains an alternate approach to understanding canonical links using convex duality. In doing so we present an improved formulation of a result on the duality of Bregman divergences that may be of independent interest.

The *Legendre-Fenchel* (LF) dual $\phi^\star$ of a function $\phi\colon \mathbb{R} \to \mathbb{R}$ is a function defined by

$$\phi^\star(s^\star) := \sup_{s\in\mathbb{R}} \{\langle s, s^\star\rangle - \phi(s)\}.$$

The LF dual of any function is convex.

When $\phi(s)$ is a function of a real argument s and the derivative $\phi'(s)$ exists, the Legendre-Fenchel conjugate $\phi^\star$ is given by the *Legendre transform* (Rockafellar, 1970; Hiriart-Urruty and Lemaréchal, 2001)

$$\phi^\star(s) = s \cdot (\phi')^{-1}(s) - \phi\left((\phi')^{-1}(s)\right). \tag{39}$$

Thus (writing $\partial f := f'$) $f' = (\partial f^\star)^{-1}$. Thus with w, W, and $\overline{W}$ defined as above,

$$W = (\partial(\overline{W}^\star))^{-1}, \quad W^{-1} = \partial(\overline{W}^\star), \ \ \overline{W}^\star = \int W^{-1}. \tag{40}$$

Let $w, W, \overline{W}$ be as in Theorem 7. Denote by L_W the w-weighted conditional loss parametrised by $W = \int w$ and let ΔL_W be the corresponding regret (we can interchange ΔL and D here by (12) since $\psi_L = \mathrm{id}$.

$$D_w(\eta, \hat{\eta}) = \overline{W}(\eta) - \overline{W}(\hat{\eta}) - (\eta - \hat{\eta})W(\hat{\eta}). \tag{41}$$

We now further consider D_w as given by (41). It will be convenient to parametrise D by W instead of w. Note that the standard parametrisation for a Bregman divergence is in terms of the convex function $\overline{W}$. Thus will write $D_{\overline{W}}, D_W$ and D_w to all represent (41). The following theorem is known (e.g., Zhang, 2004a) but as will be seen, stating it in terms of D_W provides some advantages.

Theorem 34 *Let w, W, $\overline{W}$ and D_W be as above. Then for all $x, y \in [0,1]$,*

$$D_W(x,y) = D_{W^{-1}}(W(y), W(x)). \tag{42}$$

Proof Using (39) we have

$$\begin{aligned} & \overline{W}^\star(u) = u \cdot W^{-1}(u) - \overline{W}(W^{-1}(u)) \\ \Rightarrow \quad & \overline{W}(W^{-1}(u)) = u \cdot W^{-1}(u) - \overline{W}^\star(u). \end{aligned} \tag{43}$$

Equivalently (using (40))

$$\overline{W}^\star(W(u)) = u \cdot W(u) - \overline{W}(u). \tag{44}$$

Thus substituting and then using (43) we have

$$\begin{aligned} D_W(x, W^{-1}(v)) &= \overline{W}(x) - \overline{W}(W^{-1}(v)) - (x - W^{-1}(v)) \cdot W(W^{-1}(v)) \\ &= \overline{W}(x) + \overline{W}^\star(v) - vW^{-1}(v) - (x - W^{-1}(v)) \cdot v \\ &= \overline{W}(x) + \overline{W}^\star(v) - x \cdot v. \end{aligned} \tag{45}$$

Similarly (this time using (44) we have

$$\begin{aligned} D_{W^{-1}}(v, W(x)) &= \overline{W}^\star(v) - \overline{W}^\star(W(x)) - (v - W(x)) \cdot W^{-1}(W(x)) \\ &= \overline{W}^\star(v) - xW(x) + \overline{W}(x) - v \cdot x + xW(x) \\ &= \overline{W}^\star(v) + \overline{W}(x) - v \cdot x \end{aligned} \tag{46}$$

Comparing (45) and (46) we see that

$$D_W(x, W^{-1}(v)) = D_{W^{-1}}(v, W(x))$$

Let $y = W^{-1}(v)$. Thus subsitituting $v = W(y)$ leads to (42). ■

The weight function corresponding to $D_{W^{-1}}$ is $\frac{\partial}{\partial x} W^{-1}(x) = \frac{1}{w(W^{-1}(x))}$.

Theorem 35 *If the inverse link $\psi^{-1} = W^{-1}$ (and thus $\hat{\eta} = W^{-1}(\hat{h})$) then*

$$\begin{aligned} D_W(\eta, \hat{\eta}) &= D_W(\eta, W^{-1}(\hat{h})) = \overline{W}(\eta) + \overline{W}^\star(\hat{h}) - \eta \cdot \hat{h} \\ L_W(\eta, \hat{\eta}) &= L_W(\eta, W^{-1}(\hat{h})) = \overline{W}^\star(\hat{h}) - \eta \cdot \hat{h} + \eta(\overline{W}(1) + \overline{W}(0)) - \overline{W}(0) \\ \frac{\partial}{\partial \hat{h}} L_W(\eta, W^{-1}(\hat{h})) &= \hat{\eta} - \eta \end{aligned}$$

and furthermore $D_W(\eta, W^{-1}(\hat{h}))$ and $L_W(\eta, W^{-1}(\hat{h}))$ are convex in $\hat{h}$.

Proof The first two expressions follow immediately from (45) and (46) by substitution. The derivative follows from calculation: $\frac{\partial}{\partial \hat{h}} L_W(\eta, W^{-1}(\hat{h})) = \frac{\partial}{\partial \hat{h}} (\overline{W}^\star(\hat{h}) - \eta \cdot \hat{h}) = W^{-1}(\hat{h}) - \eta = \hat{\eta} - \eta$. The convexity follows from the fact that $\overline{W}^\star$ is convex (since it is the LF dual of a convex function $\overline{W}$) and the overall expression is the sum of this and a linear term, and thus convex. ■

Buja et al. (2005) call W the *canonical link*. We have already seen (Theorem 27) that the composite loss constructed using the canonical link is convex.

Appendix C. Convexity and Robustness

In this appendix we show how the characterisation of the convexity of proper losses (Theorem 29) allows one to make general algorithm independent statements about the robustness of convex proper losses to random mis-classification noise.

Long and Servedio (2008) have shown that boosting with convex potential functions (i.e., convex margin losses) is not robust to random class noise.[15] That is, they are susceptible to random class noise. In particular they present a very simple learning task which is "boostable"—can be perfectly solved using a linear combination of base classifiers—but for which, in the presence of any amount of label noise, idealised, early stopping and L_1 regularised boosting algorithms will learn a classifier with only 50% accuracy.

This has led to the recent proposal of boosting algorithms that use non-convex margin losses and experimental evidence suggests that these are more robust to class noise than their convex counterparts. Freund (2009) recently described RobustBoost, which uses a parameterised family of non-convex surrogate losses that approximates the 0-1 loss as the number of boosting iterations increases. Experiments on a variant of the task proposed by Long and Servedio (2008) show that RobustBoost is very insensitive to class noise. Masnadi-Shirazi and Vasconcelos (2009) presented SavageBoost, a boosting algorithm built upon a non-convex margin function. They argued that even when the margin function is non-convex the conditional risk may still be convex. We elucidate this via our characterisation of the convexity of composite losses. Although all these results are suggestive, it is not clear from these results whether the robustness or not is a property of the loss function, the algorithm or a combination. We study that question by considering robustness in an algorithm-independent fashion.

For $\alpha \in (0, \frac{1}{2})$ and $\eta \in [0,1]$ we will define

$$\eta_\alpha := \alpha(1-\eta) + (1-\alpha)\eta$$

as the α-*corrupted version* of η. This captures the idea that instead of drawing a positive label for the point x with probability $\eta(x)$ there is a random class flip with probability α. This might be done on purpose in order to avoid problems with losses (e.g., log loss) that assign infinite penalty to 0 or 1 valued probability predictions. Since η_α is a convex combination of α and $1-\alpha$ it follows that $\eta_\alpha \in [\alpha, 1-\alpha]$. The effect of α-corruption on the conditional risk of a loss can be seen as a transformation of the loss (Steinwart, 2009).

Lemma 36 *If ℓ^ψ is any composite loss then its conditional risk satisfies*

$$L^\psi(\eta_\alpha, v) = L^\psi_\alpha(\eta, v), \quad \eta \in [0,1], \ \ v \in \mathcal{V},$$

where $\ell^\psi_\alpha(y,v) = (1-\alpha)\ell^\psi(y,v) + \alpha\ell^\psi(-y,v)$.

15. We define exactly what we mean by robustness below. The notion that Long and Servedio (2008) examine is akin to that studied for instance by Kearns (1998). There are many other meanings of "robust" which are different to that which we consider. The classical notion of robust statistics (Huber, 1981) is motivated by robustness to contamination of additive observation noise (some heavy-tail noise mixed in with the Gaussian noise often assumed in designing estimators). There are some results about particular machine learning algorithms being robust in that sense (Schölkopf et al., 2000). "Robust" is also used to mean robustness with respect to random attribute noise (Trafalis and Gilbert, 2006), robustness to unknown prior class probabilities (Provost and Fawcett, 2001), or a Huber-style robustness to attribute noise ("outliers") for classification (Fidler et al., 2006). We only study robustness in the sense of random label noise.

Proof By simple algebraic manipulation we have

$$\begin{aligned} L^\psi(\eta_\alpha, v) &= (1-\eta_\alpha)\ell^\psi(-1,v) + \eta_\alpha \ell^\psi(1,v) \\ &= [(1-\alpha)(1-\eta)+\alpha\eta]\ell^\psi(-1,v) + [\alpha(1-\eta)+(1-\alpha)\eta]\ell^\psi(1,v) \\ &= (1-\eta)[(1-\alpha)\ell^\psi(-1,v)+\alpha\ell^\psi(1,v)] + \eta[\alpha\ell^\psi(-1,v)+(1-\alpha)\ell^\psi(1,v)] \\ &= (1-\eta)\ell^\psi_\alpha(-1,v) + \eta\ell^\psi_\alpha(1,v) \\ &= L^\psi_\alpha(\eta, v) \end{aligned}$$

proving the result. ■

In particular, if ℓ is strictly proper then ℓ_α cannot be proper because the minimiser of $L(\eta_\alpha, \cdot)$ is η_α and so $\eta_\alpha \neq \eta$ must also be the minimiser of $L_\alpha(\eta, \cdot)$. This suggests that strictly proper losses are not robust to any class noise.

C.1 Robustness Implies Non-convexity

We now define a general notion of robustness for losses for class probability estimation.

Definition 37 *Given an* $\alpha \in [0, \frac{1}{2})$*, we will say a loss* $\ell\colon \{-1,1\} \times [0,1] \to \mathbb{R}$ *is* α-robust at η *if the set of minimisers of the conditional risk for* η *and the set of minimisers of the conditional risk for* η_α *have some common points*.

That is, a loss is α-robust for a particular η if minimising the noisy conditional risk can potentially give an estimate that is also a minimiser of the non-noisy conditional risk. Formally, ℓ *is* α*-robust at* η when

$$H(\ell, \eta_\alpha) \cap H(\ell, \eta) \neq \emptyset,$$

where $H(\ell, \eta)$ is defined in (33). Due to the equivalence of α-corruption of data and a transformed loss, another way to think about this type of robustness is the following: under what conditions can using non-proper losses still lead to the recovery of accurate conditional probability estimates?

Label noise is symmetric about $\frac{1}{2}$ and so the map $\eta \mapsto \eta_\alpha$ preserves the side of $\frac{1}{2}$ on which the values η and η_α are found. That is, $\eta \leq \frac{1}{2}$ if and only if $\eta_\alpha \leq \frac{1}{2}$ for all $\alpha \in [0, \frac{1}{2})$. This means that 0-1 misclassification loss or, equivalently, $\ell_{\frac{1}{2}}$ is α-robust for all η and for all α. For other c, the range of η for which ℓ_c is α-robust is more limited.

Theorem 38 *For each* $c \in (0,1)$*, the loss* ℓ_c *is* α*-robust at* η *if and only if*

$$\eta \notin \left[\frac{c-\alpha}{1-2\alpha}, c\right) \text{ for } c < \tfrac{1}{2} \quad \text{or} \quad \eta \notin \left[c, \frac{c-\alpha}{1-2\alpha}\right) \text{ for } c \geq \tfrac{1}{2}.$$

Proof By the definition of L_c and $[\![\hat{\eta} < c]\!] = 1 - [\![\hat{\eta} \geq c]\!]$ we have

$$L_c(\eta, \hat{\eta}) = (1-\eta)c[\![\hat{\eta} \geq c]\!] + \eta(1-c)[\![\hat{\eta} < c]\!] = \eta(1-c) + (c-\eta)[\![\hat{\eta} \geq c]\!].$$

Since $c - \eta$ is positive iff $c > \eta$ we see $L_c(\eta, \hat{\eta})$ is minimised for $\eta < c$ when $\hat{\eta} < c$ and for $\eta \geq c$ when $\hat{\eta} \geq c$. So $H(\ell_c, \eta) = [0, c)$ for $\eta < c$ and $H(\ell_c, \eta) = [c, 1]$ for $\eta \geq c$. Since $[0, c)$ and $[c, 1]$

are disjoint for all $c \in [0,1]$ we see that $H(\ell_c, \eta)$ and $H(\ell_c, \eta_\alpha)$ coincide if and only if $\eta, \eta_\alpha < c$ or $\eta, \eta_\alpha \geq c$ and are disjoint otherwise.

We proceed by cases. First, suppose $c < \frac{1}{2}$. For $\eta < c < \frac{1}{2}$ it is easy to show $\eta_\alpha \geq c$ iff $\eta \geq \frac{c-\alpha}{1-2\alpha}$ and so ℓ_c is not α-robust for $\eta \in [\frac{c-\alpha}{1-2\alpha}, c)$. For $c \leq \eta$ we see ℓ_c must be α-robust since $\eta_\alpha < c$ iff $\eta < \frac{c-\alpha}{1-2\alpha}$ but $\frac{c-\alpha}{1-2\alpha} < c$ for $c < \frac{1}{2}$ which is a contradiction. Thus, for $c < \frac{1}{2}$ we have ℓ_c is α-robust iff $\eta \notin [\frac{c-\alpha}{1-2\alpha}, c)$.

For $c > \frac{1}{2}$ the main differences are that $\frac{c-\alpha}{1-2\alpha} > c$ for $c > \frac{1}{2}$ and $\eta_\alpha < \eta$ for $\eta > \frac{1}{2}$. Thus, by a similar argument as above we see that ℓ_c is α-robust iff $\eta \notin [c, \frac{c-\alpha}{1-2\alpha})$. ■

This theorem allows us to characterise the robustness of arbitrary proper losses by appealing to the integral representation in (4).

Lemma 39 *If ℓ is a proper loss with weight function w then $H(\ell, \eta) = \bigcap_{c:\, w(c)>0} H(\ell_c, \eta)$ and so*

$$H(\ell, \eta) \cap H(\ell, \eta_\alpha) = \bigcap_{c:\, w(c)>0} H(\ell_c, \eta) \cap H(\ell_c, \eta_\alpha).$$

Proof We first show that $H(\ell, \eta) \subseteq \bigcap_{c:\, w(c)>0} H(\ell_c, \eta)$ by contradiction. Assume there is an $\hat{\eta} \in H(\ell, \eta)$ but for which there is some c_0 such that $w(c_0) > 0$ and $\hat{\eta} \notin H(\ell_{c_0}, \eta)$. Then there is a $\hat{\eta}' \in H(\ell_{c_0}, \eta)$ and $\hat{\eta}' \in H(\ell_c)$ for all other c for which $w(c) > 0$ (otherwise $H(\ell, \eta) = \{\hat{\eta}\}$). Thus, $L_{c_0}(\eta, \hat{\eta}') < L_{c_0}(\eta, \hat{\eta}')$ and so $\int_0^1 L_c(\eta, \hat{\eta}')\, w(c)\, dc < \int_0^1 L_c(\eta, \hat{\eta})\, w(c)\, dc$ since $w(c_0) > 0$.

Now suppose $\hat{\eta} \in \bigcap_{c:\, w(c)>0} H(\ell_c, \eta)$. That is, $\hat{\eta}$ is a minimiser of $L_c(\eta, \cdot)$ for all c such that $w(c) > 0$ and therefore must also be a minimiser of $L(\eta, \cdot) = \int_0^1 L_c(\eta, \cdot)\, w(c)\, dc$ and is therefore in $H(\ell, \eta)$, proving the converse. ■

One consequence of this lemma is that if $w(c) > 0$ and ℓ_c is not α-robust at η then, by definition, $H(\ell_c, \eta) \cap H(\ell_c, \eta_\alpha) = \emptyset$ and so ℓ cannot be α-robust at η. This means we have established the following theorem regarding the α-robustness of an arbitrary proper loss in terms of its weight function.

Theorem 40 *If ℓ is a proper loss with weight function w then it is not α-robust for any*

$$\eta \in \bigcup_{c:\, w(c)>0} \left[\frac{c-\alpha}{1-2\alpha}, c\right) \cup \left[c, \frac{c-\alpha}{1-2\alpha}\right).$$

By Corollary 30 we see that convex proper losses are strictly proper and thus have weight functions which are non-zero for all $c \in [0,1]$ and so by Theorem 40 we have the following corollary.

Corollary 41 *If a proper loss is convex, then for all $\alpha \in (0, \frac{1}{2})$ it is not α-robust at any $\eta \in [0,1]$.*

At a high level, this result—"convexity implies non-robustness"—appears to be logically equivalent to Long and Servedio's result that "robustness implies non-convexity". However, there are a few discrepancies that mean they are not directly comparable. The definitions of robustness differ. We focus on the point-wise minimisation of conditional risk as this is, ideally, what most risk minimisation approach try to achieve. However, this means that robustness of ERM with regularisation or restricted function classes is not directly captured with our definition whereas Long and Servedio

analyse this latter case directly. In our definition the focus is on probability estimation robustness while the earlier work is focussed on classification accuracy. Our work could be extended to this case by analysing $H(\ell,\eta)\cap H(\ell_{\frac{1}{2}},\eta)$.

Additionally, their work restricts attention to the robustness of boosting algorithms that use convex potential functions whereas our analysis is not tied to any specific algorithm. By restricting their attention to a specific learning task and class of functions they are able to show a very strong result: that convex losses for boosting lead to arbitrarily bad performance with arbitrarily little noise. Also, our focus on proper losses excludes some convex losses (such as the hinge loss) that is covered by Long and Servedio's results.

Finally, it is worth noting that there are non-convex loss functions that are strictly proper and so are not robust in the sense we use here. That is, the converse of Corollary 41 is not true. For example, any loss with weight function that sits above 0 but outside the shaded region in Figure 2 will be non-convex and non-robust. This suggests that the arguments made by Masnadi-Shirazi and Vasconcelos (2009); Freund (2009) for the robustness of non-convex losses need further investigation.

References

J.D. Abernethy, A. Agarwal, P.L. Bartlett, and A. Rakhlin. A stochastic view of optimal regret through minimax duality. March 2009. URL `http://arxiv.org/abs/0903.5328`.

J. Aczel and J. Pfanzagl. Remarks on the measurement of subjective probability and information. *Metrika*, 11(1):91–105, December 1967.

F.R. Bach, D. Heckerman, and E. Horvitz. Considering cost asymmetry in learning classifiers. *Journal of Machine Learning Research*, 7:1713–1741, 2006.

D. Bainov and P. Simeonov. *Integral Inequalities and Applications*. Kluwer, Dordrecht, 1992.

A. Banerjee, S. Merugu, I.S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

P.J. Bartlett, B. Schölkopf, D. Schuurmans, and A.J. Smola, editors. *Advances in Large-Margin Classifiers*. MIT Press, 2000.

P.L. Bartlett and A. Tewari. Sparseness vs estimating conditional probabilities: Some asymptotic results. *The Journal of Machine Learning Research*, 8:775–790, 2007.

P.L. Bartlett, M.I. Jordan, and J.D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, March 2006.

H.H. Bauschke and J.M. Borwein. Joint and separate convexity of the bregman distance. In Dan Butnariu, Yair Censor, and Simeon Reich, editors, *Inherently Parallel Algorithms in Feasibility and Optimization and their Applications*, volume 8 of *Studies in Computational Mathematics*, pages 23–36. North-Holland, 2001.

A. Beygelzimer, J. Langford, and B. Zadrozny. Machine learning techniques — reductions between prediction quality metrics. In Z. Liu and C.H. Xia, editors, *Performance Modeling and Engineering*, pages 3–28. Springer US, April 2008. URL `http://hunch.net/~jl/projects/reductions/tutorial/paper/chapter.pdf`.

A. Buja, W. Stuetzle, and Y. Shen. Loss functions for binary class probability estimation and classification: Structure and applications. Technical report, University of Pennsylvania, November 2005.

P.F. Christoffersen and F.X. Diebold. Optimal prediction under asymmetric loss. *Econometric Theory*, 13(06):808–817, 2009.

I. Cohen and M. Goldszmidt. Properties and benefits of calibrated classifiers. Technical Report HPL-2004-22(R.1), HP Laboratories, Palo Alto, July 2004.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, volume 17, pages 973–978, 2001.

S. Fidler, D. Skocaj, and A. Leonardis. Combining reconstructive and discriminative subspace methods for robust classification and regression by subsampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):337–350, 2006.

Y. Freund. A more robust boosting algorithm. arXiv:0905.2138v1 [stat.ML], May 2009. URL `http://arxiv.org/abs/0905.2138`.

T. Gneiting. Evaluating point forecasts. arXiv:0912.0902v1, December 2009.

T. Gneiting and A.E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, March 2007.

C.W.J. Granger and M.J. Machina. Forecasting and decision theory. In G. Elliot, C.W.J. Granger, and A. Timmermann, editors, *Handbook of Economic Forecasting*, volume 1, pages 82–98. North-Holland, Amsterdam, 2006.

P.D. Grünwald and A.P. Dawid. Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *The Annals of Statistics*, 32(4):1367–1433, 2004.

D.J. Hand. Deconstructing statistical questions. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, 157(3):317–356, 1994.

D.J. Hand and V. Vinciotti. Local versus global models for classification problems: Fitting models where it matters. *The American Statistician*, 57(2):124–131, 2003.

D.P. Helmbold, J. Kivinen, and M.K. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10:1291–1304, 1999.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer, Berlin, 2001.

P.J. Huber. *Robust Statistics*. Wiley, New York, 1981.

Y. Kalnishkan, V. Vovk, and M.V. Vyugin. Loss functions, complexities, and the legendre transformation. *Theoretical Computer Science*, 313(2):195–207, 2004.

Y. Kalnishkan, V. Vovk, and M.V. Vyugin. Generalised entropy and asymptotic complexities of languages. In *Learning Theory*, volume 4539/2007 of *Lecture Notes in Computer Science*, pages 293–307. Springer, 2007.

M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6): 983–1006, November 1998.

J. Kivinen and M.K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45:301–329, 2001.

D.E. Knuth. Two notes on notation. *American Mathematical Monthly*, pages 403–422, 1992.

N. Lambert, D. Pennock, and Y. Shoham. Eliciting properties of probability distributions. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 129–138, 2008.

J. Langford and B. Zadrozny. Estimating class membership probabilities using classifier learners. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT'05)*, 2005.

Y. Lin. A note on margin-based loss functions in classification. Technical Report 1044, Department of Statistics, University of Wisconsin, Madison, February 2002.

P.M. Long and R.A. Servedio. Random classification noise defeats all convex potential boosters. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *ICML*, pages 608–615, 2008. doi: 10.1145/1390156.1390233.

H. Masnadi-Shirazi and N. Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1049–1056. 2009.

P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman & Hall/CRC, 1989.

R. Nock and F. Nielsen. Bregman divergences and surrogates for learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009a. To appear.

R. Nock and F. Nielsen. On the efficient minimization of classification calibrated surrogates. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1201–1208. MIT Press, 2009b.

J. Platt. Probabilities for sv machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–71. MIT Press, 2000.

F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42 (3):203–231, 2001.

M.D. Reid and R.C. Williamson. Surrogate regret bounds for proper losses. In *Proceedings of the International Conference on Machine Learning*, pages 897–904, 2009a.

M.D. Reid and R.C. Williamson. Information, divergence and risk for binary experiments. arXiv preprint arXiv:0901.0356v1, January 2009b.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

L.J. Savage. Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 66(336):783–801, 1971.

M.J. Schervish. A general method for comparing probability assessors. *The Annals of Statistics*, 17 (4):1856–1879, 1989.

B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

Y. Shen. *Loss Functions for Binary Classification and Class Probability Estimation*. PhD thesis, Department of Statistics, University of Pennsylvania, October 2005.

E. Shuford, A. Albert, and H.E. Massengill. Admissible probability measurement procedures. *Psychometrika*, 31(2):125–145, June 1966.

C.-A. S. Staël von Holstein. *Assessment and evaluation of subjective probability distributions*. Economic Research Institute, Stockholm School of Economics, Stockholm, 1970.

I. Steinwart. How to compare different loss functions and their risks. *Constructive Approximation*, 26(2):225–287, August 2007.

I. Steinwart. Two oracle inequalities for regularized boosting classifiers. *Statistics and Its Interface*, 2:271–284, 2009.

I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, New York, 2008.

T.B. Trafalis and R.C. Gilbert. Robust classification and regression using support vector machines. *European Journal of Operational Research*, 173(3):893–909, 2006.

A. Zellner. Bayesian estimation and prediction using asymmetric loss functions. *Journal of the American Statistical Association*, 81(394):446–451, June 1986.

J. Zhang. Divergence function, duality, and convex analysis. *Neural Computation*, 16(1):159–195, 2004a.

T. Zhang. Statistical behaviour and consistency of classification methods based on convex risk minimization. *Annals of Mathematical Statistics*, 32:56–134, 2004b.

Z. Zhang, M. I. Jordan, W. J. Li, and D. Y. Yeung. Coherence functions for multicategory margin-based classification methods. In *Proceedings of the Twelfth Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Journal of Machine Learning Research 11 (2010) 2423-2455 Submitted 12/09; Published 9/10

Sparse Semi-supervised Learning Using Conjugate Functions

Shiliang Sun SHILIANGSUN@GMAIL.COM
Department of Computer Science and Technology
East China Normal University
500 Dongchuan Road, Shanghai 200241, P. R. China

John Shawe-Taylor JST@CS.UCL.AC.UK
Department of Computer Science
University College London
Gower Street, London WC1E 6BT, United Kingdom

Editor: Tony Jebara

Abstract

In this paper, we propose a general framework for sparse semi-supervised learning, which concerns using a small portion of unlabeled data and a few labeled data to represent target functions and thus has the merit of accelerating function evaluations when predicting the output of a new example. This framework makes use of Fenchel-Legendre conjugates to rewrite a convex insensitive loss involving a regularization with unlabeled data, and is applicable to a family of semi-supervised learning methods such as multi-view co-regularized least squares and single-view Laplacian support vector machines (SVMs). As an instantiation of this framework, we propose sparse multi-view SVMs which use a squared ε-insensitive loss. The resultant optimization is an inf-sup problem and the optimal solutions have arguably saddle-point properties. We present a globally optimal iterative algorithm to optimize the problem. We give the margin bound on the generalization error of the sparse multi-view SVMs, and derive the empirical Rademacher complexity for the induced function class. Experiments on artificial and real-world data show their effectiveness. We further give a sequential training approach to show their possibility and potential for uses in large-scale problems and provide encouraging experimental results indicating the efficacy of the margin bound and empirical Rademacher complexity on characterizing the roles of unlabeled data for semi-supervised learning.

Keywords: semi-supervised learning, Fenchel-Legendre conjugate, representer theorem, multi-view regularization, support vector machine, statistical learning theory

1. Introduction

Semi-supervised learning, considering how to estimate a target function from a few labeled examples and a large quantity of unlabeled examples, is one of currently active research directions. If the unlabeled data are properly used, it can get a superior performance over the counterpart supervised learning approaches. For an overview of semi-supervised learning methods, refer to Chapelle et al. (2006) and Zhu (2008).

Although semi-supervised learning was largely motivated by different real-world applications where obtaining labels is expensive or time-consuming, a lot of theoretical outcomes have also been accomplished. Typical applications of semi-supervised learning include natural image classification and text classification, where it is inexpensive to collect large numbers of images and texts by

automatic programs, but needs a high cost to label them manually. Theoretical results on semi-supervised learning include PAC-analysis (Balcan and Blum, 2005), manifold regularization (Belkin et al., 2006), and multi-view regularization theories (Sindhwani and Rosenberg, 2008), etc.

Among the methods proposed for semi-supervised learning, a family of them, for example, Laplacian regularized least squares (RLS), Laplacian support vector machines (SVMs), Co-RLS, Co-Laplacian RLS, Co-Laplacian SVMs, and manifold co-regularization (Belkin et al., 2006; Sindhwani et al., 2005; Sindhwani and Rosenberg, 2008), make use of the following representer theorem (Kimeldorf and Wahba, 1971) to represent the target function in a reproducing kernel Hilbert space (RKHS).

Theorem 1 (Representer theorem) *Let $\mathcal{H}$ be an RKHS with kernel $k : \mathcal{X} \times \mathcal{X} \to \boldsymbol{R}$. Fix any function $V : \boldsymbol{R}^n \to \boldsymbol{R}$ and any nondecreasing function $\Psi : \boldsymbol{R} \to \boldsymbol{R}$. Define*

$$J(f) = V(f(x_1), ..., f(x_n)) + \Psi(\|f\|^2),$$

and linear space $\mathcal{L} = span\{k(x_1, \cdot), ..., k(x_n, \cdot)\}$. Then for any $f \in \mathcal{H}$ we have $J(f_{\mathcal{L}}) \leq J(f)$ with $f_{\mathcal{L}}$ being the projection of f onto $\mathcal{L}$ in the following form

$$f_{\mathcal{L}} = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot).$$

Thus if $J^* = \min_f J(f)$ exists, this minimum is attained for some $f \in \mathcal{L}$. Moreover, if Ψ is strictly increasing, each minimizer of $J(f)$ over $\mathcal{H}$ must be contained in $\mathcal{L}$.

Generally, in the objective function $J(f)$ of these semi-supervised learning methods, labeled examples are used to calculate an empirical loss of the target function and simultaneously unlabeled examples are used for some regularization purpose. By the representer theorem, the target function would involve kernel evaluations on all the labeled and unlabeled examples. This is computationally undesirable, because for semi-supervised learning usually a considerably large number of unlabeled examples are available. Consequently, sparsity in the number of unlabeled data used to represent target functions is crucial, which constitutes the focus of this paper.

However, little work has been done on this theme. In particular, there is no unified framework proposed yet to deal with this sparsity concern. While the sparse Laplacian core vector machines (Tsang and Kwok, 2007) touched this problem, it has a complicated optimization and is not generic enough to generalize to other similar semi-supervised learning methods. In contrast with this, the technique developed in this paper, based on Fenchel-Legendre conjugates, is computationally simple and widely applicable.

As far as multi-view learning is concerned there has been work that introduces sparsity of the unlabeled data into the representation of the classifiers (Szedmak and Shawe-Taylor, 2007). This builds on the ideas developed for two view learning known as the SVM-2K (Farquhar et al., 2006). The approach adopted is the use of an ε-insensitive loss function for the similarity constraint between the two functions from two views. Unfortunately the resulting optimization is somewhat unmanageable and only scales to small-scale data sets despite interesting theoretical bounds that show the improvement gained using the unlabeled data.

The work by Szedmak and Shawe-Taylor (2007) forms the starting point for the current paper which aims to develop related methods that are possible to be scaled to very large data sets. Our approach is to go back to consider l_2 loss between the outputs of the classifiers arising from two

views and shows that this problem can be solved implicitly with variables only indexed by the labeled data. To compute the value of this function on new data would still require a non-sparse dual representation in terms of the unlabeled data. However, we show that through optimizing weights of the unlabeled data the solution of the l_2 problem converges to the solution of an ε-insensitive problem ensuring that we subsequently obtain sparsity in the unlabeled data. Furthermore, we develop the generalization analysis of Szedmak and Shawe-Taylor (2007) to this case giving computable expressions for the corresponding empirical Rademacher complexity.

To show the application of Fenchel-Legendre conjugates, in Section 2 we propose a novel sparse semi-supervised learning approach: sparse multi-view SVMs, where the conjugate functions play a central role in reformulating the optimization problem. The dual optimization of a subroutine of the sparse multi-view SVMs is converted to a quadratic programming problem in Section 3 whose scale only depends on the number of labeled examples, indicating the advantages of using conjugate functions. The generalization error of the sparse multi-view SVMs is given in Section 4 in terms of Rademacher complexity theory, followed by a derivation of empirical Rademacher complexity of the class of functions induced by this new method in Section 5. Section 6 reports experimental results of the sparse multi-view SVMs, comparisons with related methods, and the possibility and potential for large-scale applications through sequential training. Extensions of the use of conjugate functions to a general convex loss and other related semi-supervised learning approaches are discussed in Section 7. Finally, Section 8 concludes this paper.

2. Sparse Multi-view SVMs

Multi-view semi-supervised learning, an important branch of semi-supervised learning, combines different sets of properties of an example to learn a target function. These different sets of properties are often referred to as views. Typical applications of multi-view learning are web-page categorization and content-based multimedia information retrieval. In web-page categorization, each web-page can be simultaneously described by disparate properties such as main text, inbound and outbound hyper-links. In content-based multimedia information retrieval, a multimedia segment can include both audio and video components. For such scenarios learning with multiple views is usually very beneficial. Even for problems with no natural multiple views, artificially generated views can still work favorably (Nigam and Ghani, 2000).

A useful assumption for multi-view learning is that features from each view are sufficient to train a good learner (Blum and Mitchell, 1998; Balcan et al., 2005; Farquhar et al., 2006). Making good use of this assumption through collaborative training or regularization between views can remove many false hypotheses from the hypothesis space, and thus facilitates effective learning.

For multi-view learning, an input x consists of multiple components from different views, for example, $x = (x^1, \ldots, x^m)$ for an m-view representation. A function f_j defined on view j only depends on x^j, that is $f_j(x) := f_j(x^j)$. Suppose we have a set of ℓ labeled examples $\{(x_i, y_i)\}_{i=1}^{\ell}$ with $y_i \in \{1, -1\}$, and a set of u unlabeled examples $\{x_i\}_{i=\ell+1}^{\ell+u}$. The objective function of our sparse multi-view SVMs in the case of two views is given as follows, which can be readily extended to

more than two views.

$$\min_{f_1\in\mathcal{H}_1, f_2\in\mathcal{H}_2} \quad \frac{1}{2\ell}\sum_{i=1}^{\ell}[(1-y_i f_1(x_i))_+ + (1-y_i f_2(x_i))_+] + \gamma_n(\|f_1\|^2+\|f_2\|^2) + \gamma_v \sum_{i=1}^{\ell+u}(|f_1(x_i)-f_2(x_i)|-\varepsilon)_+^2, \tag{1}$$

where nonnegative scalars γ_n, γ_v are respectively norm regularization and multi-view regularization coefficients, and the last term is an ε-insensitive loss between two views with function $(\cdot)_+ := \max(0,\cdot)$ being the hinge loss. The final classifier for predicting the label of a new example is

$$f_c(x) = \operatorname{sgn}\left(\frac{f_1(x)+f_2(x)}{2}\right). \tag{2}$$

In the rest of this section, we will show that the use of the ε-insensitive loss indeed enforces sparsity, and Fenchel-Legendre conjugates can be adopted to reformulate the optimization problem. We also show the saddle-point properties for optimal solutions and give a (globally optimal) iterative optimization algorithm.

2.1 Sparsity

In order to show the role of the ε-insensitive loss for sparsity pursuit, here we represent $f_1(x)$ and $f_2(x)$ in feature spaces as

$$f_1(x) = \mathbf{w}_1^\top \phi_1(x) + b_1, \quad f_2(x) = \mathbf{w}_2^\top \phi_2(x) + b_2,$$

where $\phi_i(x)$ $(i=1,2)$ is the image of x in feature spaces. Problem (1) can be rewritten as

$$\begin{aligned} \min_{\mathbf{w}_1,\mathbf{w}_2,\xi_1,\xi_2,b_1,b_2} \quad & P_0 = \frac{1}{2\ell}\sum_{i=1}^{\ell}(\xi_1^i+\xi_2^i) + \gamma_n(\|\mathbf{w}_1\|^2+\|\mathbf{w}_2\|^2) + \\ & \gamma_v\sum_{i=1}^{\ell+u}(|\mathbf{w}_1^\top\phi_1(x_i)+b_1-\mathbf{w}_2^\top\phi_2(x_i)-b_2|-\varepsilon)_+^2 \\ \text{s.t.} \quad & \begin{cases} y_i(\mathbf{w}_1^\top\phi_1(x_i)+b_1) \geq 1-\xi_1^i, \\ y_i(\mathbf{w}_2^\top\phi_2(x_i)+b_2) \geq 1-\xi_2^i, \\ \xi_1^i,\ \xi_2^i \geq 0, \quad i=1,\ldots,\ell\,, \end{cases} \end{aligned}$$

where $\xi_1 := [\xi_1^1,\ldots,\xi_1^\ell]$ and $\xi_2 := [\xi_2^1,\ldots,\xi_2^\ell]$.

The Lagrangian is

$$\begin{aligned} L = \quad & P_0 - \sum_{i=1}^{\ell}[\lambda_1^i(y_i(\mathbf{w}_1^\top\phi_1(x_i)+b_1)-1+\xi_1^i) + \\ & \lambda_2^i(y_i(\mathbf{w}_2^\top\phi_2(x_i)+b_2)-1+\xi_2^i) + \nu_1^i\xi_1^i + \nu_2^i\xi_2^i], \end{aligned}$$

where $\lambda_1^i, \lambda_2^i, \nu_1^i, \nu_2^i \geq 0$ $(i=1,\ldots,\ell)$ are Lagrange multipliers.

Suppose $\mathbf{w}_{1*}, \mathbf{w}_{2*}$ are the optimal solutions. By the KKT conditions, the optimal solutions $\mathbf{w}_{1*}$ should satisfy $\frac{\partial L}{\partial \mathbf{w}_{1*}} = 0$. Therefore, we get

$$\mathbf{w}_{1*} = -\frac{\gamma_v}{\gamma_n}\sum_{i=1}^{\ell+u}(|\mathbf{w}_1^\top\phi_1(x_i)+b_1-\mathbf{w}_2^\top\phi_2(x_i)-b_2|-\varepsilon)_+\tilde{\phi}_i + \frac{1}{2\gamma_n}\sum_{i=1}^{\ell}\lambda_1^i y_i\phi_1(x_i),$$

where we suppose the derivative exists everywhere and $\tilde{\phi}_i := \text{sgn}\{\mathbf{w}_1^\top \phi_1(x_i) + b_1 - \mathbf{w}_2^\top \phi_2(x_i) - b_2\}\phi_1(x_i)$. Now we can assess the sparsity of problem (1). From the above equation, $\mathbf{w}_{1*}$ is the linear combination of labeled examples with $\lambda_1^i > 0$, and those unlabeled examples on which the difference of predictions from two views exceeds ε. In this sense, we can get sparse solutions by providing a non-zero ε. Similar analysis applies to $\mathbf{w}_{2*}$. Therefore, function $f(x)$ is sparse in the number of used unlabeled examples. This analysis on sparsity is also well justified by the representer theorem.

2.2 Reformulation Using Conjugate Functions

Define $t_i := [f_1(x_i) - f_2(x_i)]^2$. Then the ε-insensitive loss term can be written as

$$f_\varepsilon(\mathbf{t}) = \sum_{i=1}^{\ell+u} (\sqrt{t_i} - \varepsilon)_+^2, \tag{3}$$

where vector $\mathbf{t} := [t_1, \ldots, t_{\ell+u}]^\top$. We give a theorem affirming the convexity of function $f_\varepsilon(\mathbf{t})$.

Theorem 2 *Function $f_\varepsilon(\boldsymbol{t})$ defined by (3) is convex.*

Proof First, we show that $f_\varepsilon(t_i) := (\sqrt{t_i} - \varepsilon)_+^2$ with a convex domain $[0, +\infty)$ is convex. When $t_i \in (\varepsilon^2, +\infty)$, the second derivative $\nabla^2 f_\varepsilon(t_i) = \frac{1}{2}\varepsilon t_i^{-3/2} \geq 0$. Thus, function $f_\varepsilon(t_i)$ is convex for $t_i \in (\varepsilon^2, +\infty)$. Moreover, the value of function $f_\varepsilon(t_i)$ for $t_i \in (\varepsilon^2, +\infty)$ is larger than 0 which is the value of $f_\varepsilon(t_i)$ for $t_i \in [0, \varepsilon^2]$, and function $f_\varepsilon(t_i)$ with domain $[0, +\infty)$ is continuous at ε^2. Hence, $f_\varepsilon(t_i)$ is convex on the domain $[0, +\infty)$.

Then, being a nonnegative weighted sum of convex functions, $f_\varepsilon(\mathbf{t})$ is indeed convex. ■

Define conjugate vector $z = [z_1, \ldots, z_{\ell+u}]^\top$ with entries being conjugate variables. The Fenchel-Legendre conjugate (which is also often called convex conjugate or conjugate function) $f_\varepsilon^*(z)$ is

$$f_\varepsilon^*(z) = \sup_{\mathbf{t} \in \mathbf{dom} f_\varepsilon} (\mathbf{t}^\top z - f_\varepsilon(\mathbf{t})) = \sup_{\mathbf{t}} \sum_{i=1}^{\ell+u} [z_i t_i - (\sqrt{t_i} - \varepsilon)_+^2] = \sum_{i=1}^{\ell+u} \sup_{t_i} [z_i t_i - (\sqrt{t_i} - \varepsilon)_+^2].$$

The domain of the conjugate function consists of $z \in \mathbf{R}^{\ell+u}$ for which the supremum is finite (i.e., bounded above) (Boyd and Vandenberghe, 2004). Define

$$f_\varepsilon^*(z_i) = \sup_{t_i} [z_i t_i - (\sqrt{t_i} - \varepsilon)_+^2]. \tag{4}$$

Then, $f_\varepsilon^*(z) = \sum_{i=1}^{\ell+u} f_\varepsilon^*(z_i)$. As a pointwise supremum of a family of affine functions, $f_\varepsilon^*(z_i)$ is convex. Being a nonnegative weighted sum of convex functions, $f_\varepsilon^*(z)$ is also convex. Below we derive the formulation of $f_\varepsilon^*(z_i)$.

Theorem 3 *Function $f_\varepsilon^*(z_i)$ defined by (4) has the following form*

$$f_\varepsilon^*(z_i) = \begin{cases} \frac{z_i \varepsilon^2}{1 - z_i}, & \text{for } 0 < z_i < 1 \\ 0, & \text{for } z_i \leq 0\,. \end{cases} \tag{5}$$

Proof By definition, we have

$$f_\varepsilon^*(z_i) = \max_{t_i} \left\{ \sup_{0 \le t_i \le \varepsilon^2} z_i t_i, \sup_{t_i > \varepsilon^2} [z_i t_i - (\sqrt{t_i} - \varepsilon)^2] \right\}. \tag{6}$$

The value of function $\sup_{0 \le t_i \le \varepsilon^2} z_i t_i$ is simple to characterize. We now characterize the second term $\sup_{t_i > \varepsilon^2} [z_i t_i - (\sqrt{t_i} - \varepsilon)^2] = \sup_{t_i > \varepsilon^2} (z_i t_i - t_i - \varepsilon^2 + 2\varepsilon\sqrt{t_i})$. For $0 < z_i < 1$, we let the first derivative equal to zero to find the supremum. For $z_i \le 0$ or $z_i \ge 1$ the derivative does not exist and thus we use function values at end points to find the supremum. As a result, we have

$$\sup_{t_i > \varepsilon^2} [z_i t_i - (\sqrt{t_i} - \varepsilon)^2] = \begin{cases} \frac{z_i \varepsilon^2}{1 - z_i}, & \text{for } 0 < z_i < 1 \\ z_i \varepsilon^2, & \text{for } z_i \le 0 \\ +\infty, & \text{for } z_i \ge 1 \,. \end{cases}$$

According to (6) and further removing the range where $f_\varepsilon^*(z_i)$ is unbounded above, we reach the conjugate given in (5). ■

Now the Fenchel-Legendre conjugate $f_\varepsilon^*(z)$ can be represented by $\sum_{i=1}^{\ell+u} f_\varepsilon^*(z_i)$, which is also well justified by the following theorem.

Theorem 4 (Boyd and Vandenberghe, 2004) *If $\varphi(u,v) = \varphi_1(u) + \varphi_2(v)$, where φ_1 and φ_2 are independent convex functions (independent means they are functions of different variables) with conjugates φ_1^* and φ_2^*, respectively, then*

$$\varphi^*(\omega, z) = \varphi_1^*(\omega) + \varphi_2^*(z).$$

A nice property of the conjugate function is on the conjugate of the conjugate, which is central to the reformulation of our optimization problem. This property is stated by Lemma 5.

Lemma 5 (Rifkin and Lippert, 2007) *If function f is closed, convex, and proper, then the conjugate function of the conjugate is itself, that is, $f^{**} = f$, where we have defined function f is closed if its epigraph is closed, and f is proper if $\mathbf{dom} f \neq \emptyset$ and $f > -\infty$.*

It is true that function $f_\varepsilon(\mathbf{t})$ is closed and proper. Moreover, we have proved the convexity of $f_\varepsilon(\mathbf{t})$ in Theorem 2. Therefore, we can use Lemma 5 to get the following equality

$$f_\varepsilon(\mathbf{t}) = \sup_z (z^\top \mathbf{t} - f_\varepsilon^*(z)). \tag{7}$$

That is

$$\sum_{i=1}^{\ell+u} (\sqrt{t_i} - \varepsilon)_+^2 = \sup_z (z^\top \mathbf{t} - f_\varepsilon^*(z)) = \sup_z \sum_{i=1}^{\ell+u} [z_i t_i - f_\varepsilon^*(z_i)].$$

By (7), we have

$$\sum_{i=1}^{\ell+u} (|f_1(x_i) - f_2(x_i)| - \varepsilon)_+^2 = \sum_{i=1}^{\ell+u} (\sqrt{t_i} - \varepsilon)_+^2 = \sup_z \sum_{i=1}^{\ell+u} [z_i t_i - f_\varepsilon^*(z_i)].$$

Therefore, the objective function for sparse multi-view SVMs becomes

$$\min_{f_1\in\mathcal{H}_1, f_2\in\mathcal{H}_2} \quad \frac{1}{2\ell}\sum_{i=1}^{\ell}[(1-y_i f_1(x_i))_+ + (1-y_i f_2(x_i))_+] + \gamma_n(\|f_1\|^2+\|f_2\|^2) + \gamma_v \sup_z \sum_{i=1}^{\ell+u}\{z_i[f_1(x_i)-f_2(x_i)]^2 - f_\varepsilon^*(z_i)\}. \tag{8}$$

As an application of Theorem 1, the solution to problem (8) has the following form

$$f_1(x)=\sum_{i=1}^{\ell+u}\alpha_1^i k_1(x_i,x),\ f_2(x)=\sum_{i=1}^{\ell+u}\alpha_2^i k_2(x_i,x). \tag{9}$$

Applying the reproducing properties of kernels, we get

$$\|f_1\|^2=\alpha_1^\top K_1\alpha_1,\ \|f_2\|^2=\alpha_2^\top K_2\alpha_2,$$

where K_1 and K_2 are $(\ell+u)\times(\ell+u)$ Gram matrices from two views $\mathcal{V}^1$ and $\mathcal{V}^2$, respectively, and vector $\alpha_1=(\alpha_1^1,...,\alpha_1^{\ell+u})^\top$, $\alpha_2=(\alpha_2^1,...,\alpha_2^{\ell+u})^\top$. Moreover, we have

$$\mathbf{f}_1=K_1\alpha_1,\ \mathbf{f}_2=K_2\alpha_2,$$

with $\mathbf{f}_1 := (f_1(x_1),...,f_1(x_{\ell+u}))^\top$, $\mathbf{f}_2 := (f_2(x_1),...,f_2(x_{\ell+u}))^\top$. Define diagonal matrix $U=diag(z_1,\ldots,z_{\ell+u})$ with every element taking values in the range $[0,1)$. Problem (8) can be reformulated as

$$\min_{\alpha_1,\alpha_2,\xi_1,\xi_2,b_1,b_2}\sup_z \quad \frac{1}{2\ell}\sum_{i=1}^{\ell}(\xi_1^i+\xi_2^i)+\gamma_n(\alpha_1^\top K_1\alpha_1+\alpha_2^\top K_2\alpha_2)+ \gamma_v[(K_1\alpha_1-K_2\alpha_2)^\top U(K_1\alpha_1-K_2\alpha_2)-\sum_{i=1}^{\ell+u}f_\varepsilon^*(z_i)]$$
$$\text{s.t.} \quad \begin{cases} y_i(\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)\geq 1-\xi_1^i, \\ y_i(\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)+b_2)\geq 1-\xi_2^i, \\ \xi_1^i,\ \xi_2^i\geq 0, \quad i=1,\ldots,\ell\,. \end{cases} \tag{10}$$

2.3 Saddle-Point Property

We present a theorem concerning the convexity and concavity of optimization problem (10).

Theorem 6 *The objective function in problem (10) is convex with respect to* α_1,α_2, ξ_1, ξ_2, b_1, *and* b_2, *and concave with respect to* z.

Proof First, we show the convexity. The standard form of this optimization problem is

$$\min_{\alpha_1,\alpha_2,\xi_1,\xi_2,b_1,b_2}\sup_z \quad \frac{1}{2\ell}\sum_{i=1}^{\ell}(\xi_1^i+\xi_2^i)+\gamma_n(\alpha_1^\top K_1\alpha_1+\alpha_2^\top K_2\alpha_2)+ \gamma_v[(K_1\alpha_1-K_2\alpha_2)^\top U(K_1\alpha_1-K_2\alpha_2)-\sum_{i=1}^{\ell+u}f_\varepsilon^*(z_i)]$$
$$\text{s.t.} \quad \begin{cases} -y_i(\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)+1-\xi_1^i\leq 0, \\ -y_i(\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)+b_2)+1-\xi_2^i\leq 0, \\ -\xi_1^i,\ -\xi_2^i\leq 0, \quad i=1,\ldots,\ell\,. \end{cases}$$

This problem involves one objective function and three sets of inequality constraint functions (on the left hand side of each inequality). Clearly, the domain of each objective and constraint function is a convex set. Now it suffices to prove the convexity of this problem by assessing the convexity of these functions. As all constraint functions are affine, they are convex. Then, we use the second-order condition, positive semidefinite property of a function's Hessian or second derivative to judge the convexity of the objective function (Boyd and Vandenberghe, 2004). According to this condition, the first two items of the objective function are clear to be convex. The third part can be rewritten as

$$(K_1\alpha_1 - K_2\alpha_2)^\top U(K_1\alpha_1 - K_2\alpha_2) = \|U^{1/2}\begin{pmatrix} K_1 & -K_2 \end{pmatrix}\begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}\|^2,$$

which is a convex function $\|\cdot\|^2$ composed with an affine mapping and thus also convex (Boyd and Vandenberghe, 2004). Being a nonnegative weighted sums of convex functions, the objective function is therefore convex with respect to $\alpha_1, \alpha_2, \xi_1, \xi_2, b_1, b_2$.

Then, we show the concavity using (8). As $f_\varepsilon^*(z_i)$ is convex, $z_i[f_1(x_i) - f_2(x_i)]^2 - f_\varepsilon^*(z_i)$ is concave with respect to z_i. The concavity of $\sum_{i=1}^{\ell+u}\{z_i[f_1(x_i) - f_2(x_i)]^2 - f_\varepsilon^*(z_i)\}$ follows from the fact that a nonnegative weighted sum of concave functions is concave (Boyd and Vandenberghe, 2004). Hence, the objective function in problem (10) is concave with respect to z. ■

Let θ denote the parameters $\alpha_1, \alpha_2, \xi_1, \xi_2, b_1, b_2$. We can simply denote the above optimization problem as

$$\inf_\theta \sup_z f(\theta, z) \tag{11}$$

associated with constraints on the labeled examples, where $f(\theta, z)$ is convex with respect to θ, and concave with respect to z. We give the following theorem on the equivalence of swapping the infimum and supremum for our optimization problem and include a proof for completeness.

Theorem 7 (Boyd and Vandenberghe, 2004) *If $f(\theta, \boldsymbol{z})$ with domain Θ and Z is convex with respect to $\theta \in \Theta$, and concave with respect to $\boldsymbol{z} \in Z$, the following equality holds*

$$\inf_\theta \sup_{\boldsymbol{z}} f(\theta, \boldsymbol{z}) = \sup_{\boldsymbol{z}} \inf_\theta f(\theta, \boldsymbol{z})$$

under some slight assumptions.

Proof The idea is first to represent the left-hand side as a value of a convex function, and then show that the conjugate of its conjugate is equal to the right-hand side when the same input value is plugged in Boyd and Vandenberghe (2004).

The left-hand side can be expressed as $p(0)$, where

$$p(\mathbf{u}) = \inf_\theta \sup_z [f(\theta, z) + \mathbf{u}^\top z] .$$

It is not difficult to show that p is a convex function. Being a pointwise supremum of convex function $f(\theta, z) + \mathbf{u}^\top z$, $\sup_z[f(\theta, z) + \mathbf{u}^\top z]$ is a convex function of $(\theta, \mathbf{u})$. Because $\sup_z[f(\theta, z) + \mathbf{u}^\top z]$ is convex with respect to $(\theta, \mathbf{u})$, we have $p(\mathbf{u})$ is convex.

The Fenchel-Legendre conjugate of $p(\mathbf{u})$ is

$$p^*(\mathbf{v}) = \sup_{\mathbf{u}}[\mathbf{u}^\top \mathbf{v} - \inf_\theta \sup_z (f(\theta, z) + \mathbf{u}^\top z)] ,$$

which would be $+\infty$ if $z \neq \mathbf{v}$. Therefore,

$$p^*(\mathbf{v}) = \begin{cases} -\inf_\theta f(\theta, \mathbf{v}), & \text{for } \mathbf{v} \in Z \\ +\infty, & \text{otherwise .} \end{cases}$$

The conjugate of $p^*(z)$ is given by

$$p^{**}(\mathbf{u}) = \sup_{z \in Z}(\mathbf{u}^\top z - p^*(z)) = \sup_{z \in Z}(\mathbf{u}^\top z + \inf_\theta f(\theta, z)) = \sup_z \inf_\theta [f(\theta, z) + \mathbf{u}^\top z] .$$

Suppose $0 \in \mathbf{dom} p(\mathbf{u})$ and $p(\mathbf{u})$ is closed and proper. Then by Lemma 5 we have $p(0) = p^{**}(0)$ which completes the proof. ■

Now we give a theorem showing that the optimal pair $\tilde{\theta}, \tilde{z}$ is a saddle-point.

Theorem 8 *If the following equality holds for function $f(\theta, z)$*

$$\inf_\theta \sup_z f(\theta, z) = \sup_z \inf_\theta f(\theta, z) = f(\tilde{\theta}, \tilde{z}) ,$$

then the optimal pair $\tilde{\theta}, \tilde{z}$ forms a saddle-point.

Proof From the given equality, we have

$$\inf_\theta \sup_z f(\theta, z) = \sup_z f(\tilde{\theta}, z) = f(\tilde{\theta}, \tilde{z}) ,$$

and

$$\sup_z \inf_\theta f(\theta, z) = \inf_\theta f(\theta, \tilde{z}) = f(\tilde{\theta}, \tilde{z}) ,$$

Therefore,

$$f(\tilde{\theta}, z) \leq f(\tilde{\theta}, \tilde{z}) \leq f(\theta, \tilde{z}),$$

which indeed satisfies the definition of a saddle-point. The proof is completed. ■

2.4 Iterative Optimization Algorithm

To solve the optimization problem $\sup_z \inf_\theta f(\theta, z)$ which is respectively concave and convex with respect to z and θ, we give an algorithm with guaranteed convergence by the following theorem.

Theorem 9 *Given an initial value z_0 for z, solve $\inf_\theta f(\theta, z_0)$ and obtain the global optimal point θ_0. Then we find $\arg\max_z f(\theta_0, z)$ to get z_1 from which we can get θ_1 as a result of optimize $\inf_\theta f(\theta, z_1)$. Repeat this process until a convergence point $(\hat{\theta}, \hat{z})$ is reached. Suppose $\tilde{\theta}, \tilde{z}$ is a saddle point. We have $f(\hat{\theta}, \hat{z}) = f(\tilde{\theta}, \tilde{z})$. That is, we got the optimal values of the objective function. If f is strictly concave and strictly convex with respect to the variables, we further have $\hat{\theta} = \tilde{\theta}$ and $\hat{z} = \tilde{z}$.*

Proof According to the properties of function f and the algorithm procedure, we know that the convergence point is a saddle point. Thus, we have

$$f(\tilde{\theta}, \hat{z}) \geq f(\hat{\theta}, \hat{z}) \geq f(\hat{\theta}, \tilde{z}) .$$

By the saddle-point property of $(\tilde{\theta},\tilde{z})$, we have

$$f(\tilde{\theta},\tilde{z}) \geq f(\tilde{\theta},\hat{z}) ,$$

and

$$f(\tilde{\theta},\tilde{z}) \leq f(\hat{\theta},\tilde{z}) .$$

Therefore, the above inequalities should hold with equalities and we have $f(\tilde{\theta},\tilde{z}) = f(\hat{\theta},\hat{z})$. Furthermore, if f is strictly concave and strictly convex with respect to the variables, it is true that $\hat{\theta} = \tilde{\theta}$ and $\hat{z} = \tilde{z}$. ■

On solving $\arg\max_{\mathbf{z}} f(\theta,\mathbf{z})$ required in Theorem 9, we can maximize the term related to $\mathbf{z}$, namely $z^\top \mathbf{t} - f_\varepsilon^*(z) = \sum_{i=1}^{\ell+u}[z_i t_i - f_\varepsilon^*(z_i)]$. For this purpose, we have the following theorem.

Theorem 10

$$\sup_{z_i \in \mathbf{dom} f_\varepsilon^*(z_i)} [z_i t_i - f_\varepsilon^*(z_i)] = (\sqrt{t_i} - \varepsilon)_+^2,$$

and

$$\arg \sup_{z_i \in \mathbf{dom} f_\varepsilon^*(z_i)} [z_i t_i - f_\varepsilon^*(z_i)] = \begin{cases} 1 - \frac{\varepsilon}{\sqrt{t_i}}, & \text{for } t_i > \varepsilon^2 \\ 0, & \text{for } 0 \leq t_i \leq \varepsilon^2 . \end{cases}$$

Without loss of generality, we can confine the range of z_i to $[0,1)$.

Proof We have

$$\sup_{z_i \in \mathbf{dom} f_\varepsilon^*(z_i)} [z_i t_i - f_\varepsilon^*(z_i)] = \max_{z_i} \{ \sup_{0<z_i<1} z_i t_i - \frac{z_i \varepsilon^2}{1-z_i}, \sup_{z_i \leq 0} z_i t_i \} .$$

The first supremum can be solved by setting the derivative with respect to z_i to zero. We have

$$\sup_{0<z_i<1} z_i t_i - \frac{z_i \varepsilon^2}{1-z_i} = (\sqrt{t_i} - \varepsilon)^2$$

where $t_i > \varepsilon^2$, and the supremum is attained with $z_i = 1 - \frac{\varepsilon}{\sqrt{t_i}}$.

When $t_i < 0$, $\sup_{z_i \leq 0} z_i t_i$ is unbounded above. When $0 \leq t_i \leq \varepsilon^2$, $\sup_{z_i \leq 0} z_i t_i = 0$ with the supremum attained at $z_i = 0$. Therefore, $\max_{z_i}\{\sup_{0<z_i<1} z_i t_i - \frac{z_i \varepsilon^2}{1-z_i}, \sup_{z_i \leq 0} z_i t_i\} = (\sqrt{t_i} - \varepsilon)_+^2$ with the supremum attained when $z_i \in [0,1)$, which completes the proof. ■

For sparsity pursuit, during each iteration we remove those unlabeled examples whose corresponding z_i's are zero. By the representer theorem, this would not influence the value of the objective function. For Theorem 9, this means that the element of z whose values are zero in the last iteration will remain zero for the next iteration. When there are no unlabeled examples eligible for elimination, the iteration will terminate and the convergence point $(\hat{\theta},\hat{\mathbf{z}})$ is reached.

3. Dual Optimization

According to the iterative optimization algorithm, when optimizing problem (10), we start from an initial value z_0 and then solve θ_0. In this section, we show how to solve this subroutine with fixed z.

Now the optimization problem is equivalent to

$$\begin{aligned}\min_{\alpha_1,\alpha_2,\xi_1,\xi_2,b_1,b_2} \quad & F_0 = \frac{1}{2\ell}\sum_{i=1}^{\ell}(\xi_1^i+\xi_2^i)+\gamma_n(\alpha_1^\top K_1\alpha_1+\alpha_2^\top K_2\alpha_2)+ \\ & \gamma_v(K_1\alpha_1-K_2\alpha_2)^\top U(K_1\alpha_1-K_2\alpha_2) \\ \text{s.t.} \quad & \begin{cases} y_i(\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)\geq 1-\xi_1^i, \\ y_i(\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)+b_2)\geq 1-\xi_2^i, \\ \xi_1^i,\ \xi_2^i\geq 0, \quad i=1,\ldots,\ell\,. \end{cases}\end{aligned} \tag{12}$$

3.1 Lagrange Dual Function

We will solve problem (12) through optimizing its dual problem which is simpler to solve. Now we derive its Lagrange dual function.

Suppose $\lambda_1^i,\lambda_2^i,\nu_1^i,\nu_2^i\geq 0$ $(i=1,\ldots,\ell)$ be the Lagrange multipliers associated with the inequality constraints. Define $\lambda_j=[\lambda_j^1,\ldots,\lambda_j^\ell]^\top$ and $\nu_j=[\nu_j^1,\ldots,\nu_j^\ell]^\top$ $(j=1,2)$. The Lagrangian $L(\alpha_1,\alpha_2,\xi_1,\xi_2,b_1,b_2,\lambda_1,\lambda_2,\nu_1,\nu_2)$ can be written as

$$\begin{aligned}L= \quad & F_0-\sum_{i=1}^{\ell}[\lambda_1^i(y_i(\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)-1+\xi_1^i)+ \\ & \lambda_2^i(y_i(\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)+b_2)-1+\xi_2^i)+\nu_1^i\xi_1^i+\nu_2^i\xi_2^i].\end{aligned}$$

Note that

$$\begin{aligned}& (K_1\alpha_1-K_2\alpha_2)^\top U(K_1\alpha_1-K_2\alpha_2) \\ = \quad & \alpha_1^\top K_1UK_1\alpha_1-2\alpha_1^\top K_1UK_2\alpha_2+\alpha_2^\top K_2UK_2\alpha_2.\end{aligned}$$

To obtain the Lagrangian dual function, L has to be minimized with respect to the primal variables $\alpha_1,\alpha_2,\xi_1,\xi_2,b_1,b_2$. To eliminate these variables, we compute the corresponding partial derivatives and set them to 0, obtaining the following conditions

$$2J_1\alpha_1-2\gamma_v K_1UK_2\alpha_2=\Lambda_1, \tag{13}$$

$$2J_2\alpha_2-2\gamma_v K_2UK_1\alpha_1=\Lambda_2, \tag{14}$$

$$\lambda_1^i+\nu_1^i=\frac{1}{2\ell}, \tag{15}$$

$$\lambda_2^i+\nu_2^i=\frac{1}{2\ell}, \tag{16}$$

$$\sum_{i=1}^{\ell}\lambda_1^i y_i=0,$$

$$\sum_{i=1}^{\ell}\lambda_2^i y_i=0, \tag{17}$$

where

$$\begin{aligned} J_1 &:= \gamma_n K_1 + \gamma_v K_1 U K_1, \\ J_2 &:= \gamma_n K_2 + \gamma_v K_2 U K_2, \\ \Lambda_1 &:= \sum_{i=1}^{\ell} \lambda_1^i y_i K_1(:,i), \\ \Lambda_2 &:= \sum_{i=1}^{\ell} \lambda_2^i y_i K_2(:,i), \end{aligned}$$

with $K_1(:,i)$ and $K_2(:,i)$ being the ith column of the corresponding Gram matrices.

Substituting (13)$\sim$(17) into L results in the following expression of the Lagrangian dual function $g_L(\lambda_1, \lambda_2, \nu_1, \nu_2)$

$$\begin{aligned} g_L &= \gamma_n(\alpha_1^\top K_1 \alpha_1 + \alpha_2^\top K_2 \alpha_2) + \gamma_v(\alpha_1^\top K_1 U K_1 \alpha_1 - 2\alpha_1^\top K_1 U K_2 \alpha_2 + \\ &\quad \alpha_2^\top K_2 U K_2 \alpha_2) - \alpha_1^\top \Lambda_1 - \alpha_2^\top \Lambda_2 + \sum_{i=1}^{\ell} (\lambda_1^i + \lambda_2^i) \\ &= \frac{1}{2}\alpha_1^\top \Lambda_1 + \frac{1}{2}\alpha_2^\top \Lambda_2 - \alpha_1^\top \Lambda_1 - \alpha_2^\top \Lambda_2 + \sum_{i=1}^{\ell} (\lambda_1^i + \lambda_2^i) \\ &= -\frac{1}{2}\alpha_1^\top \Lambda_1 - \frac{1}{2}\alpha_2^\top \Lambda_2 + \sum_{i=1}^{\ell} (\lambda_1^i + \lambda_2^i). \end{aligned} \tag{18}$$

We obtain the following from (13) and (14)

$$\alpha_1 = \frac{1}{2} J_1^{-1}(\Lambda_1 + 2\gamma_v K_1 U K_2 \alpha_2) \tag{19}$$

$$\alpha_2 = \frac{1}{2} J_2^{-1}(\Lambda_2 + 2\gamma_v K_2 U K_1 \alpha_1). \tag{20}$$

From (13) and (20), we have

$$(2J_1 - 2\gamma_v^2 K_1 U K_2 J_2^{-1} K_2 U K_1)\alpha_1 = \Lambda_1 + \gamma_v K_1 U K_2 J_2^{-1} \Lambda_2.$$

Define $M_1 = 2J_1 - 2\gamma_v^2 K_1 U K_2 J_2^{-1} K_2 U K_1$. Suppose the above linear system is well-posed (if ill-posed we can employ approximate numerical analysis techniques). We get

$$\alpha_1 = M_1^{-1}(\Lambda_1 + \gamma_v K_1 U K_2 J_2^{-1} \Lambda_2).$$

From (14) and (19), we have

$$(2J_2 - 2\gamma_v^2 K_2 U K_1 J_1^{-1} K_1 U K_2)\alpha_2 = \Lambda_2 + \gamma_v K_2 U K_1 J_1^{-1} \Lambda_1.$$

Define $M_2 = 2J_2 - 2\gamma_v^2 K_2 U K_1 J_1^{-1} K_1 U K_2$. Thus we get

$$\alpha_2 = M_2^{-1}(\Lambda_2 + \gamma_v K_2 U K_1 J_1^{-1} \Lambda_1).$$

Now with α_1 and α_2 substituted into (18), the Lagrange dual function $g_L(\lambda_1,\lambda_2,\nu_1,\nu_2)$ is

$$\begin{aligned} g_L &= \inf_{\alpha_1,\alpha_2,\xi_1,\xi_2,b_1,b_2} L = -\frac{1}{2}\alpha_1^\top \Lambda_1 - \frac{1}{2}\alpha_2^\top \Lambda_2 + \sum_{i=1}^{\ell}(\lambda_1^i+\lambda_2^i) \\ &= -\frac{1}{2}(\Lambda_1+\gamma_v K_1 U K_2 J_2^{-1}\Lambda_2)^\top M_1^{-1}\Lambda_1 - \frac{1}{2}(\Lambda_2 + \\ &\quad \gamma_v K_2 U K_1 J_1^{-1}\Lambda_1)^\top M_2^{-1}\Lambda_2 + \sum_{i=1}^{\ell}(\lambda_1^i+\lambda_2^i). \end{aligned}$$

3.2 Solving the Dual Problem

The Lagrange dual problem is given by

$$\begin{aligned} \max_{\lambda_1,\lambda_2} \quad & g_L \\ \text{s.t.} \quad & \begin{cases} 0 \le \lambda_1^i \le \frac{1}{2\ell}, & i=1,\ldots,\ell \\ 0 \le \lambda_2^i \le \frac{1}{2\ell}, & i=1,\ldots,\ell \\ \sum_{i=1}^{\ell}\lambda_1^i y_i = 0, & \\ \sum_{i=1}^{\ell}\lambda_2^i y_i = 0. & \end{cases} \end{aligned}$$

As Lagrange dual functions are always concave (Boyd and Vandenberghe, 2004), we can formulate the above problem as a convex optimization problem

$$\begin{aligned} \min_{\lambda_1,\lambda_2} \quad & -g_L \\ \text{s.t.} \quad & \begin{cases} 0 \le \lambda_1^i \le \frac{1}{2\ell}, & i=1,\ldots,\ell \\ 0 \le \lambda_2^i \le \frac{1}{2\ell}, & i=1,\ldots,\ell \\ \sum_{i=1}^{\ell}\lambda_1^i y_i = 0, & \\ \sum_{i=1}^{\ell}\lambda_2^i y_i = 0. & \end{cases} \end{aligned} \tag{21}$$

Define matrix $Y = diag(y_1,\ldots,y_\ell)$. Then, $\Lambda_1 = K_{\ell 1} Y \lambda_1$ and $\Lambda_2 = K_{\ell 2} Y \lambda_2$ with $K_{\ell 1} = K_1(:,1:\ell)$ and $K_{\ell 2} = K_2(:,1:\ell)$. We have

$$\begin{aligned} -g_L &= \frac{1}{2}(\Lambda_1+\gamma_v K_1 U K_2 J_2^{-1}\Lambda_2)^\top M_1^{-1}\Lambda_1 + \frac{1}{2}(\Lambda_2 + \\ &\quad \gamma_v K_2 U K_1 J_1^{-1}\Lambda_1)^\top M_2^{-1}\Lambda_2 - \sum_{i=1}^{\ell}(\lambda_1^i+\lambda_2^i) \\ &= \frac{1}{2}(\lambda_1^\top \lambda_2^\top)\begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} - \mathbf{1}^\top(\lambda_1+\lambda_2), \end{aligned}$$

where

$$\begin{aligned} A &:= Y K_{\ell 1}^\top M_1^{-1} K_{\ell 1} Y, \\ B &:= \gamma_v Y K_{\ell 1}^\top J_1^{-1} K_1 U K_2 M_2^{-1} K_{\ell 2} Y, \\ C &:= \gamma_v Y K_{\ell 2}^\top J_2^{-1} K_2 U K_1 M_1^{-1} K_{\ell 1} Y, \\ D &:= Y K_{\ell 2}^\top M_2^{-1} K_{\ell 2} Y, \end{aligned}$$

and $\mathbf{1} = (1, \ldots, 1_{(\ell)})^\top$.

Substituting M_1 and M_2 into the expressions of B and C, we can prove that $B = C^\top$. In addition, because of the convexity of function $-g$, we affirm that matrix $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$ is positive semi-definite.

Hence, the optimization problem in (21) can be rewritten as

$$\begin{aligned} \min_{\lambda_1,\lambda_2} \quad & \frac{1}{2}(\lambda_1^\top \lambda_2^\top)\begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} - \mathbf{1}^\top(\lambda_1+\lambda_2) \\ \text{s.t.} \quad & \begin{cases} 0 \preceq \lambda_1 \preceq \frac{1}{2l}\mathbf{1}, \\ 0 \preceq \lambda_2 \preceq \frac{1}{2l}\mathbf{1}, \\ \lambda_1^\top \mathbf{y} = 0, \\ \lambda_2^\top \mathbf{y} = 0, \end{cases} \end{aligned}$$

where $\mathbf{y} = (y_1, \ldots, y_\ell)^\top$. After solving this problem using standard software, we then obtain ν_1^i and ν_2^i by (15) and (16).

We now state the advantages of optimizing this dual problem over optimizing the primal problem (12):

- Less optimization variables as for typical semi-supervised learning $\ell \ll u$, and
- Simpler constraint functions.

The solution of bias terms b_1 and b_2 can be obtained through support vectors. Due to KKT conditions, the following equalities hold

$$\begin{aligned} &\lambda_1^i(y_i(\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)-1+\xi_1^i)=0, \\ &\lambda_2^i(y_i(\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)+b_2)-1+\xi_2^i)=0, \\ &\nu_1^i\xi_1^i=0, \\ &\nu_2^i\xi_2^i=0, \quad i=1,\ldots,\ell. \end{aligned}$$

For support vectors x_i, we have $\nu_j^i > 0$ (and thus $\xi_j^i = 0$) and $\lambda_j^i > 0$ $(j = 1,2)$. Therefore, we can resolve the bias terms by averaging $y_i(\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)-1=0$ and $y_i(\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)+b_2)-1=0$ over all support vectors.

3.3 Advantages of Using Conjugate Functions

In this subsection, we show the direct optimization of problem (1) without the use of conjugate functions is of large scale and time-consuming, which justifies the advantages of using conjugate functions.

The primal problem can be rewritten as

$$\min_{\alpha_1,\alpha_2,\xi_1,\xi_2,b_1,b_2,\delta_i} D_0 = \frac{1}{2\ell}\sum_{i=1}^{\ell}(\xi_1^i+\xi_2^i)+\gamma_n(\alpha_1^\top K_1\alpha_1+\alpha_2^\top K_2\alpha_2)+\gamma_v\sum_{i=1}^{\ell+u}\delta_i^2$$

$$\text{s.t.}\begin{cases} y_i(\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)\geq 1-\xi_1^i, \quad i=1,\ldots,\ell\,, \\ y_i(\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)+b_2)\geq 1-\xi_2^i, \quad i=1,\ldots,\ell\,, \\ \xi_1^i,\ \xi_2^i\geq 0, \quad i=1,\ldots,\ell\,, \\ (\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)-(\sum_{j=1}^{l+u}\alpha_2^j k_2(x_j,x_i)+b_2)\geq -\delta_i-\varepsilon, \quad i=1,\ldots,\ell+u\,, \\ (\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)-(\sum_{j=1}^{l+u}\alpha_2^j k_2(x_j,x_i)+b_2)\leq \delta_i+\varepsilon, \quad i=1,\ldots,\ell+u\,, \end{cases} \tag{22}$$

where $y_i\in\{1,-1\}, \gamma_n,\gamma_v\geq 0$.

We will solve problem (22) through optimizing its dual problem which can be simpler to solve. Suppose $\lambda_1^i,\lambda_2^i,\nu_1^i,\nu_2^i\geq 0$ $(i=1,\ldots,\ell)$ and μ_1^i,μ_2^i $(i=1,\ldots,\ell+u)$ are the Lagrange multipliers associated with the inequality constraints of problem (22). Define $\delta=[\delta_1,\ldots,\delta_{\ell+u}]^\top, \lambda_j=[\lambda_j^1,\ldots,\lambda_j^\ell]^\top$, $\nu_j=[\nu_j^1,\ldots,\nu_j^\ell]^\top$, and $\mu_j=[\mu_j^1,\ldots,\mu_j^{\ell+u}]^\top$ $(j=1,2)$. The Lagrangian $L(\alpha_1,\alpha_2,\xi_1,\xi_2,b_1,b_2,\delta,\lambda_1,\lambda_2,\nu_1,\nu_2,\mu_1,\mu_2)$ can be written as

$$\begin{aligned} L= \quad & D_0-\sum_{i=1}^{\ell}[\lambda_1^i(y_i(\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1)-1+\xi_1^i)+ \\ & \lambda_2^i(y_i(\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)+b_2)-1+\xi_2^i)+\nu_1^i\xi_1^i+\nu_2^i\xi_2^i]- \\ & \sum_{i=1}^{\ell+u}\mu_1^i[\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1-\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)-b_2+\delta_i+\varepsilon]+ \\ & \sum_{i=1}^{\ell+u}\mu_2^i[\sum_{j=1}^{\ell+u}\alpha_1^j k_1(x_j,x_i)+b_1-\sum_{j=1}^{\ell+u}\alpha_2^j k_2(x_j,x_i)-b_2-\delta_i-\varepsilon]. \end{aligned}$$

To obtain the Lagrangian dual function, L has to be minimized with respect to the primal variables $\alpha_1,\alpha_2,\xi_1,\xi_2,b_1,b_2,\delta$. To eliminate these variables, we compute the corresponding partial

derivatives and set them to 0, obtaining the following conditions

$$
\begin{aligned}
&2\gamma_n K_1\alpha_1 = \sum_{i=1}^{\ell}\lambda_1^i y_i K_1(:,i) + \sum_{i=1}^{\ell+u}(\mu_1^i - \mu_2^i)K_1(:,i),\\
&2\gamma_n K_2\alpha_2 = \sum_{i=1}^{\ell}\lambda_2^i y_i K_2(:,i) - \sum_{i=1}^{\ell+u}(\mu_1^i - \mu_2^i)K_2(:,i),\\
&\lambda_1^i + \nu_1^i = \frac{1}{2\ell}, \quad i = 1,\ldots,\ell\\
&\lambda_2^i + \nu_2^i = \frac{1}{2\ell}, \quad i = 1,\ldots,\ell\\
&-\sum_{i=1}^{\ell}\lambda_1^i y_i - \sum_{i=1}^{\ell+u}\mu_1^i + \sum_{i=1}^{\ell+u}\mu_2^i = 0,\\
&-\sum_{i=1}^{\ell}\lambda_2^i y_i + \sum_{i=1}^{\ell+u}\mu_1^i - \sum_{i=1}^{\ell+u}\mu_2^i = 0,\\
&2\gamma_v\delta_i - \mu_1^i - \mu_2^i = 0, \quad i = 1,\ldots,\ell+u.
\end{aligned}
$$

Substituting these equations into the Lagrangian as what was done in Section 3.1, it is clear that finally L is a quadratic function involving $\lambda_1,\lambda_2,\mu_1,\mu_2$. The dual optimization problem would be a quadratic optimization involving $2\ell+2(\ell+u)$ parameters. Now we see this direct optimization is indeed of large-scale and time-consuming.

4. Generalization Error

In this section, we analyze the generalization performance of the sparse multi-view SVMs making use of Rademacher complexity theory and the margin bound.

4.1 Rademacher Complexity Theory

Important background on Rademacher complexity theory (Bartlett and Mendelson, 2002; Shawe-Taylor and Cristianini, 2004) is introduced below.

Definition 11 *For a sample $S = \{x_1,\ldots,x_\ell\}$ generated by a distribution $\mathcal{D}$ on a set $\mathcal{X}$ and a real-valued function class $\mathcal{F}$ with domain $\mathcal{X}$, the empirical Rademacher complexity of $\mathcal{F}$ is the random variable*

$$\hat{R}_\ell(\mathcal{F}) = \mathbb{E}_\sigma\left[\sup_{f\in\mathcal{F}}\left|\frac{2}{\ell}\sum_{i=1}^{\ell}\sigma_i f(x_i)\right| \Big| x_1,\ldots,x_\ell\right],$$

where $\sigma = \{\sigma_1,\ldots,\sigma_\ell\}$ are independent uniform $\{\pm1\}$-valued (Rademacher) random variables. The Rademacher complexity of $\mathcal{F}$ is

$$R_\ell(\mathcal{F}) = \mathbb{E}_S[\hat{R}_\ell(\mathcal{F})] = \mathbb{E}_{S\sigma}\left[\sup_{f\in\mathcal{F}}\left|\frac{2}{\ell}\sum_{i=1}^{\ell}\sigma_i f(x_i)\right|\right].$$

Lemma 12 *Fix $\delta\in(0,1)$ and let $\mathcal{F}$ be a class of functions mapping from an input space $\tilde{\mathcal{X}}$ ($\tilde{\mathcal{X}} = \mathcal{X}\times\mathcal{Y}$ or $\tilde{\mathcal{X}} = \mathcal{X}$) to $[0,1]$. Let $(\tilde{x}_i)_{i=1}^{\ell}$ be drawn independently according to a probability distribution*

$\mathcal{D}$. Then with probability at least $1-\delta$ over random draws of samples of size ℓ, every $f \in \mathcal{F}$ satisfies

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[f(\tilde{x})] &\leq \hat{\mathbb{E}}[f(\tilde{x})] + R_\ell(\mathcal{F}) + \sqrt{\frac{\ln(2/\delta)}{2\ell}} \\ &\leq \hat{\mathbb{E}}[f(\tilde{x})] + \hat{R}_\ell(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}},\end{aligned}$$

where $\hat{\mathbb{E}}[f(\tilde{x})]$ is the empirical error averaged on the ℓ examples.

4.2 Margin Bound for Sparse Multi-view SVMs

By (2), the prediction function of the sparse multi-view SVMs is derived from the average of predictions from two views. Define the soft prediction function as

$$g(x) = \frac{1}{2}(f_1(x) + f_2(x)).$$

We obtain the following margin bound regarding the generalization error of sparse multi-view SVMs. This bound is widely applicable to multi-view SVMs, for example, Szedmak and Shawe-Taylor (2007) independently provided a similar bound for the SVM-2K method.

Theorem 13 *Fix $\delta \in (0,1)$ and let $\mathcal{F}$ be the class of functions mapping from $\tilde{\mathcal{X}} = \mathcal{X} \times \mathcal{Y}$ to $\boldsymbol{R}$ given by $\tilde{f}(x,y) = -yg(x)$ where $g = \frac{1}{2}(f_1 + f_2) \in \mathcal{G}$ and $\tilde{f} \in \mathcal{F}$. Let $S = \{(x_1,y_1), \cdots, (x_\ell, y_\ell)\}$ be drawn independently according to a probability distribution $\mathcal{D}$. Then with probability at least $1-\delta$ over samples of size ℓ, every $g \in \mathcal{G}$ satisfies*

$$P_{\mathcal{D}}(y \neq sgn(g(\boldsymbol{x}))) \leq \quad \frac{1}{2\ell}\sum_{i=1}^{\ell}(\xi_1^i + \xi_2^i) + 2\hat{R}_\ell(\mathcal{G}) + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

where $\xi_1^i := (1 - y_i f_1(x_i))_+$, $\xi_2^i := (1 - y_i f_2(x_i))_+$. Function $y_i f_1(x_i)$ and $y_i f_2(x_i)$ are called margins.

Proof Let $H(\cdot)$ be the Heaviside function that returns 1 if its argument is greater than 0 and zero otherwise. We have

$$P_{\mathcal{D}}(y \neq sgn(g(\mathbf{x}))) = \mathbb{E}_{\mathcal{D}}[H(-yg(\mathbf{x}))]. \tag{23}$$

Consider a loss function $\mathcal{A} : \mathbf{R} \to [0,1]$, given by

$$\mathcal{A}(a) = \begin{cases} 1, & \text{if } a \geq 0; \\ 1+a, & \text{if } -1 \leq a \leq 0; \\ 0, & \text{otherwise.} \end{cases}$$

By Lemma 12 and since function $\mathcal{A} - 1$ dominates $H - 1$, we have (Shawe-Taylor and Cristianini, 2004)

$$\begin{aligned}&\mathbb{E}_{\mathcal{D}}[H(\tilde{f}(x,y)) - 1] \leq \mathbb{E}_{\mathcal{D}}[\mathcal{A}(\tilde{f}(x,y)) - 1] \\ &\leq \hat{\mathbb{E}}[\mathcal{A}(\tilde{f}(x,y)) - 1] + \hat{R}_\ell((\mathcal{A} - 1) \circ \mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}.\end{aligned}$$

Therefore,

$$\mathbb{E}_{\mathcal{D}}[H(\tilde{f}(x,y))] \leq \quad \hat{\mathbb{E}}[\mathcal{A}(\tilde{f}(x,y))] + \hat{R}_\ell((\mathcal{A}-1)\circ\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}.$$

In addition, we have

$$\begin{aligned}
\hat{E}[\mathcal{A}(\tilde{f}(x,y))] &\leq \frac{1}{\ell}\sum_{i=1}^{\ell}(1-y_i g(x_i))_+ \\
&= \frac{1}{2\ell}\sum_{i=1}^{\ell}(1-y_i f_1(x_i)+1-y_i f_2(x_i))_+ \\
&\leq \frac{1}{2\ell}\sum_{i=1}^{\ell}[(1-y_i f_1(x_i))_+ + (1-y_i f_2(x_i))_+] \\
&= \frac{1}{2\ell}\sum_{i=1}^{\ell}(\xi_1^i + \xi_2^i),
\end{aligned}$$

where ξ_1^i denotes the amount by which function f_1 fails to achieve margin 1 for (x_i, y_i) and ξ_2^i applies similarly to function f_2.

Since $(\mathcal{A}-1)(0) = 0$, we can apply the Lipschitz condition (Bartlett and Mendelson, 2002) of function $(\mathcal{A}-1)$ to get

$$\hat{R}_\ell((\mathcal{A}-1)\circ\mathcal{F}) \leq 2\hat{R}_\ell(\mathcal{F}).$$

It remains to bound the empirical Rademacher complexity of the class $\mathcal{F}$.

With $y_i \in \{1,-1\}$, we have

$$\begin{aligned}
\hat{R}_\ell(\mathcal{F}) &= \mathbb{E}_\sigma[\sup_{f\in\mathcal{F}}|\frac{2}{\ell}\sum_{i=1}^{\ell}\sigma_i\tilde{f}(x_i,y_i)|] \\
&= \mathbb{E}_\sigma[\sup_{g\in\mathcal{G}}|\frac{2}{\ell}\sum_{i=1}^{\ell}\sigma_i y_i g(x_i)|] \\
&= \mathbb{E}_\sigma[\sup_{g\in\mathcal{G}}|\frac{2}{\ell}\sum_{i=1}^{\ell}\sigma_i g(x_i)|] \\
&= \hat{R}_\ell(\mathcal{G}). \qquad (24)
\end{aligned}$$

Finally, combining (23)∼(24) completes the proof. ■

5. Empirical Rademacher Complexity

Our optimization algorithm iteratively updates z to solve θ. In this section, we first derive the empirical Rademacher complexity of $\hat{R}_\ell(\mathcal{G})$ for the function class induced after one iteration with an initially fixed z, and then give its formulation applicable for any number of subsequent iterations including the termination case. This Rademacher complexity is crucial for Theorem 13 when analyzing the performance of the corresponding classifiers obtained by our iterative optimization algorithm. Specifically, for the empirical Rademacher complexity we give the following theorem

Theorem 14 *Suppose* $\mathcal{S} = \frac{1}{\gamma_n}(K_{1\ell}K_1^{-1}K_{1\ell}^\top + K_{2\ell}K_2^{-1}K_{2\ell}^\top)$, $\Theta = \frac{1}{\gamma_n}U_u^{1/2}(K_{1u}K_1^{-1}K_{1u}^\top + K_{2u}K_2^{-1}K_{2u}^\top)U_u^{1/2}$, $\mathcal{J} = \frac{1}{\gamma_n}U_u^{1/2}(K_{1u}K_1^{-1}K_{1\ell}^\top - K_{2u}K_2^{-1}K_{2\ell}^\top)$, *where* $K_{1\ell}$ *and* $K_{2\ell}$ *are respectively the first* ℓ *rows of the Gram matrices* K_1 *and* K_2, K_{1u} *and* K_{2u} *are respectively the last* u *rows of matrix* K_1 *and* K_2, *and* U_u *is the diagonal matrix including the last* u *diagonal elements (initially fixed* $z_{\ell+1},\ldots,z_{\ell+u}$*) of* U. *Then the empirical Rademacher complexity* $\hat{R}_\ell(\mathcal{G})$ *is bounded as* $\frac{\mathcal{U}}{\sqrt{2\ell}} \le \hat{R}_\ell(\mathcal{G}) \le \frac{\mathcal{U}}{\ell}$, *where* $\mathcal{U}^2 = tr(\mathcal{S}) - \gamma_v tr(\mathcal{J}^\top(I+\gamma_v\Theta)^{-1}\mathcal{J})$ *for the first iteration of sparse multi-view SVMs, and* $\mathcal{U}^2 = tr(\mathcal{S})$ *for subsequent iterations*.

The remainder of this section before Section 5.4 completes the proof of this theorem, which was partially inspired by Rosenberg and Bartlett (2007) for analyzing co-regularized least squares.

We use problem (8) to reason about $\hat{R}_\ell(\mathcal{G})$. As a result of fixed z, we can remove $f_\varepsilon^*(z_i)$ without loss of generality to resolve f_1 and f_2. It is true that the loss function $\hat{L} : \mathcal{H}^1 \times \mathcal{H}^2 \to [0,\infty)$ with $\hat{L} := \frac{1}{2\ell}\sum_{i=1}^{\ell}[(1-y_i f_1(x_i))_+ + (1-y_i f_2(x_i))_+]$ satisfies

$$\hat{L}(0,0) = 1.$$

Let $Q(f_1,f_2)$ denote the objective function in (8) with $f_\varepsilon^*(z_i)$ removed. Substituting in the trivial predictors $f_1 \equiv 0$ and $f_2 \equiv 0$ gives the following upper bound

$$\min_{f_1,f_2\in\mathcal{H}^1\times\mathcal{H}^2} Q(f_1,f_2) \le Q(0,0) = \hat{L}(0,0) = 1.$$

Since all terms of $Q(f_1,f_2)$ are nonnegative, we conclude that any (f_1^*,f_2^*) minimizing $Q(f_1,f_2)$ is contained in

$$\hat{\mathcal{H}} \;=\; \{(f_1,f_2) : \gamma_n(\|f_1\|^2 + \|f_2\|^2) + \gamma_v \sum_{i=\ell+1}^{\ell+u} z_i[f_1(x_i) - f_2(x_i)]^2 \le 1\}. \tag{25}$$

Therefore, the final predictor is chosen from the function class

$$\mathcal{G} = \{x \to \frac{1}{2}[f_1(x) + f_2(x)] : (f_1,f_2) \in \hat{\mathcal{H}}\}.$$

The complexity $\hat{R}_\ell(\mathcal{G})$ is

$$\hat{R}_\ell(\mathcal{G}) = \mathbb{E}_\sigma\left[\sup_{(f_1,f_2)\in\hat{\mathcal{H}}}\left|\frac{1}{\ell}\sum_{i=1}^{\ell}\sigma_i(f_1(x_i)+f_2(x_i))\right|\right]. \tag{26}$$

As it only depends on the values of function $f_1(\cdot)$ and $f_2(\cdot)$ on the ℓ labeled examples, by the reproducing kernel property which says the projection of function f onto a closed subspace containing $k(x,\cdot)$ has the same value at x as f itself does (Rosenberg and Bartlett, 2007) we can restrict the function class $\hat{\mathcal{H}}$ to the span of labeled and unlabeled data and thus write it as

$$\begin{aligned}\hat{\mathcal{H}} \;&=\; \{(f_1,f_2) : \gamma_n(\alpha_1^\top K_1\alpha_1 + \alpha_2^\top K_2\alpha_2) + \\ &\quad\; \gamma_v(K_{1u}\alpha_1 - K_{2u}\alpha_2)^\top U_u(K_{1u}\alpha_1 - K_{2u}\alpha_2) \le 1\} \\ &=\; \{(f_1,f_2) : (\alpha_1^\top\;\alpha_2^\top)N\begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} \le 1\},\end{aligned}$$

where K_{1u} and K_{2u} are respectively the last u rows of matrix K_1 and K_2, U_u is the diagonal matrix including the last u diagonal elements of U, and

$$N := \gamma_n\begin{pmatrix}K_1 & \mathbf{0}\\ \mathbf{0} & K_2\end{pmatrix} + \gamma_v\begin{pmatrix}K_{1u}^\top\\ -K_{2u}^\top\end{pmatrix}U_u(K_{1u}\;\; -K_{2u}). \tag{27}$$

5.1 Evaluating the Supremum in Euclidean Space

Since $(f_1, f_2) \in \hat{\mathcal{H}}$ implies $(-f_1, -f_2) \in \hat{\mathcal{H}}$, we can drop the absolute sign in (26). Now we can write

$$\begin{aligned}\hat{R}_\ell(\mathcal{G}) &= \frac{1}{\ell}\mathbb{E}_\sigma \sup_{\alpha_1,\alpha_2\in\mathcal{R}^{\ell+u}} \{\sigma^\top K_{1\ell}\alpha_1 + \sigma^\top K_{2\ell}\alpha_2 : (\alpha_1^\top\ \alpha_2^\top)N\begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} \le 1\} \\ &= \frac{1}{\ell}\mathbb{E}_\sigma \sup_{\alpha_1,\alpha_2\in\mathcal{R}^{\ell+u}} \{\sigma^\top (K_{1\ell}\ K_{2\ell})\begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} : (\alpha_1^\top\ \alpha_2^\top)N\begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} \le 1\}, \qquad (28)\end{aligned}$$

where $K_{1\ell}, K_{2\ell}$ represent the first ℓ rows of the Gram matrices K_1 and K_2, respectively.

For a symmetric positive definite matrix M, it is simple to show that (Rosenberg and Bartlett, 2007)

$$\sup_{\alpha:\alpha^\top M\alpha\le 1} \mathbf{v}^\top\alpha = \|M^{-1/2}\mathbf{v}\|.$$

Without loss of generality, suppose positive semi-definite matrix N in (28) is positive definite and thus has full rank. If N does not have full rank, we can use subspace decomposition to rewrite $\hat{R}_\ell(\mathcal{G})$ to obtain a similar representation. Thus, we can evaluate the supremum as described above to get

$$\hat{R}_\ell(\mathcal{G}) = \frac{1}{\ell}\mathbb{E}_\sigma\|N^{-1/2}\begin{pmatrix}K_{1\ell}^\top\\ K_{2\ell}^\top\end{pmatrix}\sigma\|.$$

5.2 Bounding $\hat{R}_\ell(\mathcal{G})$ above and below

We make use of the Kahane-Khintchine inequality (Latala and Oleszkiewicz, 1994), stated here for convenience, to bound $\hat{R}_\ell(\mathcal{G})$.

Lemma 15 *For any vectors $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n$ in a Hilbert space and independent Rademacher random variables $\sigma_1, \cdots, \sigma_n$, we have*

$$\frac{1}{2}\mathbb{E}\|\sum_{i=1}^n \sigma_i\boldsymbol{a}_i\|^2 \le (\mathbb{E}\|\sum_{i=1}^n \sigma_i\boldsymbol{a}_i\|)^2 \le \mathbb{E}\|\sum_{i=1}^n \sigma_i\boldsymbol{a}_i\|^2.$$

By Lemma 15 we have

$$\frac{\mathcal{U}}{\sqrt{2}\ell} \le \hat{R}_\ell(\mathcal{G}) \le \frac{\mathcal{U}}{\ell}, \qquad (29)$$

where

$$\begin{aligned}\mathcal{U}^2 &= \mathbb{E}_\sigma\|N^{-1/2}\begin{pmatrix}K_{1\ell}^\top\\ K_{2\ell}^\top\end{pmatrix}\sigma\|^2 \\ &= \mathbb{E}_\sigma tr[(K_{1\ell}\ K_{2\ell})N^{-1}\begin{pmatrix}K_{1\ell}^\top\\ K_{2\ell}^\top\end{pmatrix}\sigma\sigma^\top] \\ &= tr[(K_{1\ell}\ K_{2\ell})N^{-1}\begin{pmatrix}K_{1\ell}^\top\\ K_{2\ell}^\top\end{pmatrix}].\end{aligned}$$

Recall that

$$N = \gamma_n\begin{pmatrix}K_1 & \mathbf{0}\\ \mathbf{0} & K_2\end{pmatrix} + \gamma_v\begin{pmatrix}K_{1u}^\top\\ -K_{2u}^\top\end{pmatrix}U_u(K_{1u}\ \ -K_{2u}).$$

Define

$$\Sigma = \gamma_n \begin{pmatrix} K_1 & \mathbf{0} \\ \mathbf{0} & K_2 \end{pmatrix}, \quad R = \begin{pmatrix} K_{1u}^\top \\ -K_{2u}^\top \end{pmatrix} U_u^{1/2}.$$

Using the Sherman-Morrison-Woodbury formula (Golub and Loan, 1996), we expand N^{-1} as

$$N^{-1} = \Sigma^{-1} - \gamma_v \Sigma^{-1} R (I + \gamma_v R^\top \Sigma^{-1} R)^{-1} R^\top \Sigma^{-1}.$$

Define $\Omega = (K_{1\ell}\ K_{2\ell})$. We get

$$\mathcal{U}^2 = tr(\Omega\Sigma^{-1}\Omega^\top) - \gamma_v tr[\Omega\Sigma^{-1} R (I + \gamma_v R^\top \Sigma^{-1} R)^{-1} R^\top \Sigma^{-1} \Omega^\top].$$

Define

$$\begin{aligned} \mathcal{S} &= \Omega\Sigma^{-1}\Omega^\top = \frac{1}{\gamma_n}(K_{1\ell}K_1^{-1}K_{1\ell}^\top + K_{2\ell}K_2^{-1}K_{2\ell}^\top), \\ \Theta &= R^\top\Sigma^{-1}R = \frac{1}{\gamma_n}U_u^{1/2}(K_{1u}K_1^{-1}K_{1u}^\top + K_{2u}K_2^{-1}K_{2u}^\top)U_u^{1/2}, \\ \mathcal{J} &= R^\top\Sigma^{-1}\Omega^\top = \frac{1}{\gamma_n}U_u^{1/2}(K_{1u}K_1^{-1}K_{1\ell}^\top - K_{2u}K_2^{-1}K_{2\ell}^\top). \end{aligned} \tag{30}$$

Putting expressions together, we get

$$\mathcal{U}^2 = tr(\mathcal{S}) - \gamma_v tr(\mathcal{J}^\top (I + \gamma_v \Theta)^{-1} \mathcal{J}). \tag{31}$$

5.2.1 REGULARIZATION TERM ANALYSIS

From (29) and (31), it is clear to see the roles the regularization parameters γ_n and γ_v play in the empirical Rademacher complexity $\hat{R}_l(\mathcal{G})$.

The amount of reduction in the Rademacher complexity brought by γ_v is

$$\Delta(\gamma_v) = \gamma_v tr(\mathcal{J}^\top (I + \gamma_v \Theta)^{-1} \mathcal{J}).$$

This term has the property shown by the following lemma given by Rosenberg and Bartlett (2007) when analyzing co-regularized least squares. Here the meanings of $\mathcal{J}$ and Θ are different from Rosenberg and Bartlett (2007).

Lemma 16 (Rosenberg and Bartlett, 2007) *$\Delta(0) = 0$, $\Delta(\gamma_v)$ is nondecreasing on $\gamma_v \geq 0$, and given that Θ is positive definite, we have*

$$\lim_{\gamma_v \to \infty} \Delta(\gamma_v) = tr(\mathcal{J}^\top \Theta^{-1} \mathcal{J}).$$

5.3 Extending to Iterative Optimization

As our sparse multi-view SVMs employ an iterative optimization procedure for sparsity pursuit, the former outcome for empirical Rademacher complexity would not apply if we use more than one iteration to update z. However, we can extend the former analysis to suit this case.

Recall that $z_i \in [0,1)$ $(i = \ell+1,\ldots,\ell+u)$ and $U_u = diag(z_{\ell+1},\ldots,z_{\ell+u})$. During iterations, it is possible that U_u becomes a zero matrix or other arbitrary matrix with diagonal elements in the range $[0,1)$. In any case, the resultant function class can be covered by

$$\hat{\mathcal{H}} \quad = \quad \{(f_1,f_2) : \gamma_n(\|f_1\|^2 + \|f_2\|^2) \leq 1\},$$

which is obtained by omitting the term containing z_i in (25). Following a similar derivation, the matrix N in (27) would be

$$N = \gamma_n \begin{pmatrix} K_1 & \mathbf{0} \\ \mathbf{0} & K_2 \end{pmatrix}.$$

Finally, we can obtain a bound on the empirical Rademacher complexity $\hat{R}_l(\mathcal{G})$ identical to (29) but now $\mathcal{U}^2 = tr(\mathcal{S})$ with $\mathcal{S}$ defined in (30). The proof of Theorem 14 is completed.

5.4 Examining $\hat{R}_l(\mathcal{G})$

Here, we examine the role $\hat{R}_l(\mathcal{G})$ plays in the margin bound. Since $K_{1\ell}$ and $K_{2\ell}$ are the first ℓ rows of K_1 and K_2, the formulation of $tr(\mathcal{S})$ can be simplified as

$$tr(\mathcal{S}) \quad = \quad \frac{1}{\gamma_n} tr(K_{1\ell} K_1^{-1} K_{1\ell}^\top + K_{2\ell} K_2^{-1} K_{2\ell}^\top) = \frac{1}{\gamma_n} \sum_{i=1}^{\ell} (K_1(i,i) + K_2(i,i)). \tag{32}$$

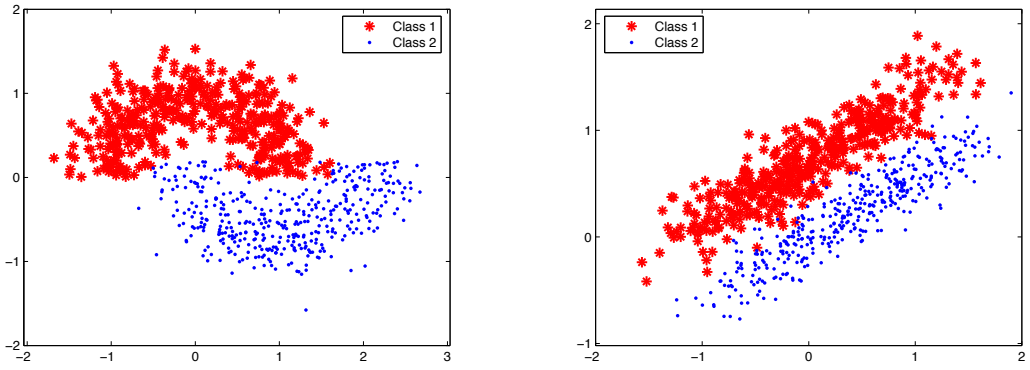
Now, we see that for iterative optimization of sparse multi-view SVMs, the empirical Rademacher complexity $\hat{R}_l(\mathcal{G})$ with $\mathcal{U}^2 = tr(\mathcal{S})$ only depends on the ℓ labeled examples and the chosen kernel functions. Consequently, the margin bound does not rely on the unlabeled training sets. In this case the margin bound is quite straightforward to reason.

If we do not use iterative optimization, the empirical Rademacher complexity $\hat{R}_l(\mathcal{G})$ will involve other two terms Θ and $\mathcal{J}$. By a similar technique as in (32), we can show that Θ only depends on the unlabeled data and the kernel functions, while $\mathcal{J}$ encodes the interaction between labeled and unlabeled data. As a result, the margin bound relies on both labeled and unlabeled data. For this case, we will give an evaluation of the margin bound with different sizes of unlabeled sets in Section 6.4.

6. Experiments

We performed experiments on artificial data and real-world data to evaluate the proposed sparse multi-view SVMs (SpMvSVMs). For SpMvSVMs with $\varepsilon > 0$, the entries of $\mathbf{z}$ were fixed as 1 for labeled data and initialized as 0.995 for unlabeled data. The termination condition for iterative optimization is either no unlabeled examples can be removed or the maximum iteration number surpasses 50. Comparisons are made with supervised SVMs, and the unsupervised SVM-2K method. Each accuracy/error reported in this paper is an averaged accuracy/error value over ten random splits of data into labeled, unlabeled and test data.

Later in this section, we also provide a sequential training strategy for SpMvSVMs, which shows an accuracy improvement over the gradual adding of unlabeled data while with roughly linear and sub-linear increases of running time. This indicates the possibility and potential of applying SpMvSVMs to large-scale data sets. At the end, margin bound evaluation results are reported.



(a) View 1 (b) View 2

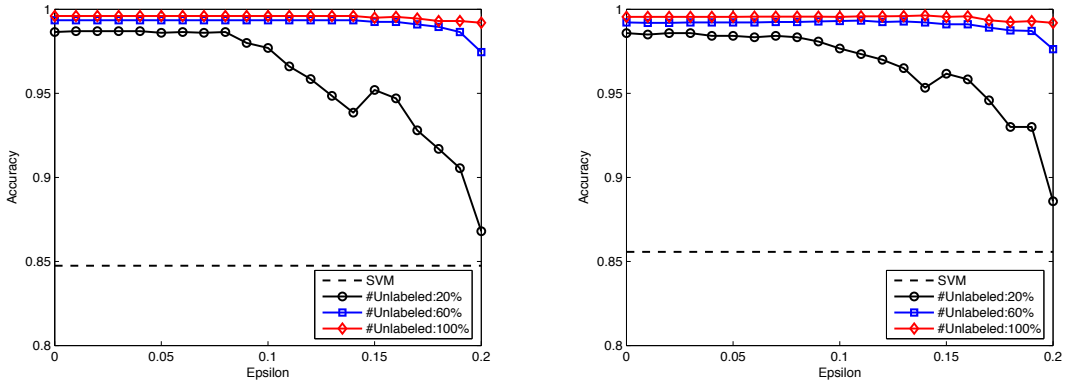
Figure 1: Examples in the two-moons-two-lines data set.

6.1 Artificial Data

This two-moons-two-lines synthetic data set was generated according to Sindhwani et al. (2005). Examples in two classes scatter like two moons in one view and two parallel lines in the other. To link the two views, points on one moon were enforced to associate at random with points on one line. Each class has 400 examples and a total of 800 examples were generated as shown in Figure 1. For SpMvSVMs, the numbers of examples in the labeled training set, unlabeled training set and test set were fixed as four, 596, and 200, respectively. Gaussian kernel with bandwidth 0.35 and the linear kernel were used for view 1 and view 2, respectively. The parameters γ_n and γ_v were selected from a small grid $\{10^{-6}, 10^{-4}, 10^{-2}, 1, 10, 100\}$ by five-fold cross validation on the whole data set. The chosen values are $\gamma_n = 10^{-4}$ and $\gamma_v = 1$. In this paper, γ_v is normalized by the number of labeled and unlabeled examples involved in the multi-view regularization term. For supervised SVMs, which concatenated features from the two views, we also found the regularization coefficient from this grid by five-fold cross validation.

To evaluate SpMvSVMs, we varied the size of the unlabeled training set from 20%, 60% to 100% of the total number of unlabeled data, and used different values for the insensitive parameter ε, which ranged from 0 to 0.2 with an interval 0.01 (when ε is zero, sparsity is not considered). The test accuracies and transductive accuracies (on the corresponding unlabeled set) are given in Figure 2(a) and Figure 2(b), respectively. It should be noted that the numbers of data used to calculate transductive accuracies are different for the three curves in Figure 2(b). The numbers of removed unlabeled examples for different ε values are shown in Figure 3.

From Figure 2 and Figure 3, we find that with the increase of ε, more and more unlabeled data are removed, and the remove of a small number of unlabeled data can hardly decrease the performance of the resultant classifiers, especially when the original size of unlabeled set is large. Therefore, we can find a good balance between sparsity and accuracy using an appropriate ε. In addition, more unlabeled data can benefit the performance of the learned classifiers with the same ε.



(a) Test accuracy (b) Transductive accuracy

Figure 2: Classification accuracies of SpMvSVMs with different sizes of unlabeled set and ε values on the artificial data. The accuracies of SVMs are also shown.

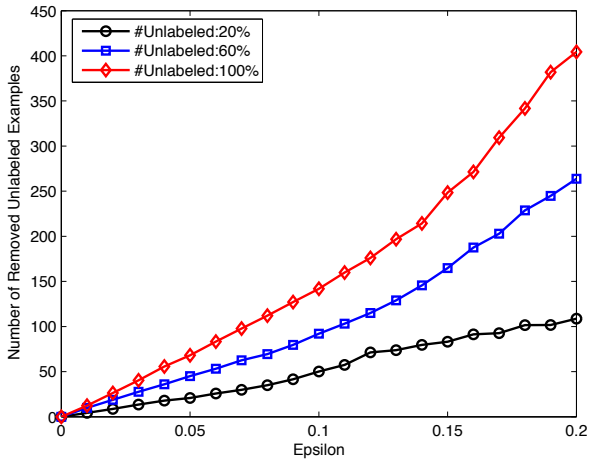
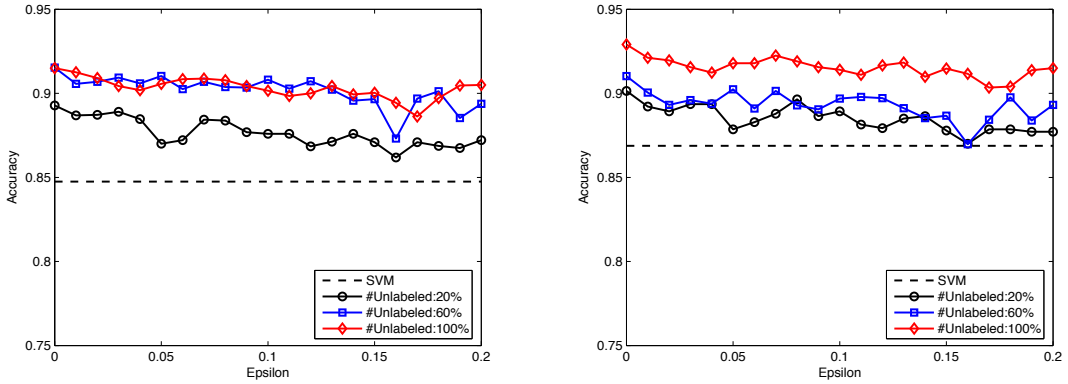


Figure 3: The numbers of unlabeled examples removed by SpMvSVMs for different ε values on the artificial data.

6.2 Text Classification

We applied the SpMvSVMs to the WebKB text classification task studied in Blum and Mitchell (1998); Sindhwani et al. (2005); Sun (2008). The data set consists of 1051 two-view web pages collected from the computer science department web sites at four U.S. universities: Cornell, University of Washington, University of Wisconsin, and University of Texas. The task is to predict whether a web page is a course home page or not. This problem has an unbalanced class distribution since there are a total of 230 course home pages (positive examples). The first view of the data is the words appearing on the web page itself, whereas the second view is the underlined words



(a) Test accuracy

(b) Transductive accuracy

Figure 4: Classification accuracies of SpMvSVMs with different sizes of unlabeled set and ε values on text classification. The accuracies of SVMs are also shown.

in all links pointing to the web page from other pages. We preprocessed each view by removing stop words, punctuation and numbers and then applied Porter's stemming to the text (Porter, 1980). In addition, words that occur in five or fewer documents were ignored. This resulted in 2332 and 87-dimensional vectors in the first and second view, respectively. Finally, document vectors were normalized to $tf.idf$ (the product of term frequency and inverse document frequency) features (Salton and Buckley, 1988).

For SpMvSVMs, the numbers of examples in the labeled training set, unlabeled training set and test set were fixed as 32, 699, and 320, respectively. In the training set and test set, the numbers of negative examples are three times of those of positive examples to reflect the overall proportion of positive and negative examples. The linear kernel was used for both views. The parameters γ_n and γ_v for SpMvSVMs and the regularziation coefficient for SVMs were selected using the same method as in Section 6.1. The chosen values for SpMvSVMs are $\gamma_n = 10^{-6}$ and $\gamma_v = 0.01$.

To evaluate SpMvSVMs, we also varied the size of the unlabeled training set from 20%, 60% to 100% of the total number of unlabeled data, and used different values for the insensitive parameter ε ranging from 0 to 0.2 with an interval 0.01. The test accuracies and transductive accuracies are given in Figure 4(a) and Figure 4(b), respectively. The numbers of removed unlabeled examples for different ε values are shown in Figure 5.

From Figure 5, we find that with the increase of ε, more and more unlabeled data can be removed. Reflected by Figure 4, the remove of unlabeled data only slightly decrease the performance of the resultant classifiers, and this decrease is less when more unlabeled data is used. We draw a same conclusion as before: an appropriate ε can be adopted to keep a good balance between sparsity and accuracy. We also observe a different phenomenon, that is, using 60% and 100% unlabeled data result in similar test accuracies as shown in Figure 4(a). This is reasonable because the performance improvement of any classifier is always bounded no matter how many data are used.

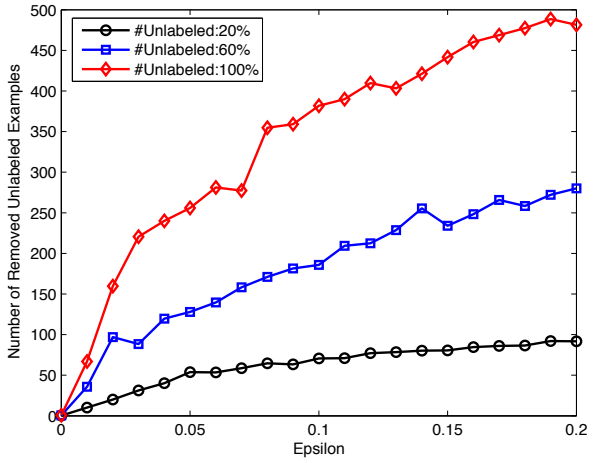


Figure 5: The numbers of unlabeled examples removed by SpMvSVMs for different ε values on text classification.

6.3 Comparison with SVM-2K, and Sequential Training

The SVM-2K method proposed by Szedmak and Shawe-Taylor (2007) can exploit unlabeled data for multi-view learning. Similar to SpMvSVMs, it also combines the maximum margin and multi-view regularization principles. However, it adopts l_1 norm for multi-view regularization, and this regularization only uses unlabeled data. Specifically, the SVM-2K method has the following optimization for classifier parameters $\mathbf{w}_1$, $\mathbf{w}_2$, b_1, and b_2 in two views

$$\begin{aligned}
\min \quad & \frac{1}{2}\|\mathbf{w}_1\|^2+\frac{1}{2}\|\mathbf{w}_2\|^2+C_1\sum_{i=1}^{\ell}\xi_1^i+C_2\sum_{i=1}^{\ell}\xi_2^i+C_\eta\sum_{j=\ell+1}^{\ell+u}\eta_j \\
\text{s.t.} \quad & \begin{cases} |\mathbf{w}_1^\top\phi_1(x_j)+b_1-\mathbf{w}_2^\top\phi_2(x_j)-b_2|\le\eta_j+\varepsilon \\ y_i(\mathbf{w}_1^\top\phi_1(x_i)+b_1)\ge 1-\xi_1^i \\ y_i(\mathbf{w}_2^\top\phi_2(x_i)+b_2)\ge 1-\xi_2^i \\ \xi_1^i\ge 0,\xi_2^i\ge 0,\eta_j\ge 0 \text{ for all } i=1,\ldots,\ell, \text{and } j=\ell+1,\ldots,\ell+u\,, \end{cases}
\end{aligned}$$

where an ε-insensitive parameter is used to relax the prediction consistency between views. In this subsection, we carry out an empirical comparison between SVM-2K and our SpMvSVMs for semi-supervised learning with identical data splits.

Our first comparison takes the ε-insensitive parameter in both SpMvSVMs and SVM-2K as zero and uses the above two data sets with different sizes of unlabeled training sets, namely, from 20% to 60% to 100%. For SVM-2K, we adopted the same parameter selection approach as in Szedmak and Shawe-Taylor (2007) through five-fold cross-validation. That is, the values of C_1 and C_2 were fixed to 1 and C_η were selected from the range $\{0.01\times 2^i\}$ $(i=1,\ldots,10)$. The experimental results are listed in Table 1, from which we see that both the test accuracies and transductive accuracies of SpMvSVMs are superior to those counterparts of SVM-2K.

The second comparison considers sequential training of SpMvSVMs and SVM-2K. The purpose is to show the relationship between running time, classification accuracies and the number of

# Unlabeled	SVM-2K		SpMvSVMs	
	Test Acc.	Transductive Acc.	Test Acc.	Transductive Acc.
20%	95.70	97.75	98.65	98.58
60%	98.50	99.19	99.35	99.22
100%	98.35	99.18	99.60	99.55
20%	84.72	84.71	89.28	90.14
60%	85.88	84.79	91.53	91.02
100%	85.84	87.85	91.50	92.90

Table 1: Test and transductive accuracies (%) of SVM-2K and SpMvSVMs with different sizes of unlabeled training sets on the artificial data (the first three lines) and text classification data (the last three lines).

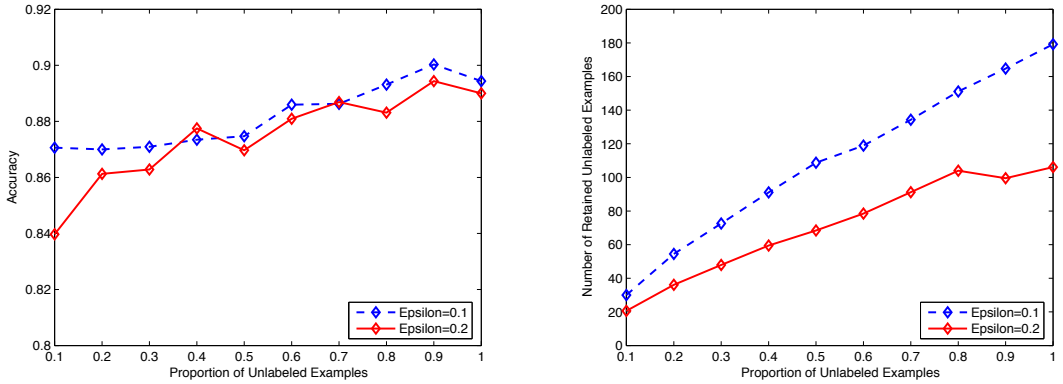
gradually added unlabeled examples, and thus evaluate the possibility and potential of applying the methods to large-scale problems. The text classification data were used where all the unlabeled training data were divided into ten equal sizes. For sequential training of SpMvSVMs, we adopted two different ε values 0.1 and 0.2. The procedure is as follows. First, we train SpMvSVMs using the labeled data and the first portion of unlabeled data. Then, we combine the retained unlabeled data from the last training with the next portion of unlabeled data together to train SpMvSVMs (with the original labeled data). We repeat this progress for ten times to complete the whole procedure.

The test accuracies and total numbers of retained unlabeled data after each step are shown in Figure 6(a) and Figure 6(b). The averaged classifier training time is given in Figure 7. Figure 6(a) indicates the effectiveness of sequential training, which is reflected by the fact that the test accuracies have an overall increasing tendency. Figure 6(b) shows that SpMvSVMs obtain sparse solutions in the sense that the number of retained unlabeled data is small compared to the number of all the added unlabeled data. Figure 7 shows that when $\varepsilon = 0.1$ the running time is roughly linear with respect to the gradual adding of unlabeled data, and when $\varepsilon = 0.2$ the relationship is roughly sub-linear. In fact, the running time can be further reduced if larger ε values rather than the given values are properly used. In practice, we can also vary the value of ε during the sequential training process.

Though the SVM-2K method was not initially proposed for sparse semi-supervised learning, we find that with $\varepsilon > 0$ it can reduce the number of unlabeled data used for representing classifiers. For this reason, we attempted to explore its possibility on sequential training using this data set under the same setting with the sequential training of SpMvSVMs. However, for different ε values we did not observe an improvement of test accuracies with the gradual adding of unlabeled data. Actually, for this data set when $\varepsilon > 0.05$ SVM-2K would not use any unlabeled data at all. This indicates that the roles of ε for SpMvSVMs and SVM-2K are quantitatively different.

6.4 Margin Bound Evaluation

To evaluate the margin bound in Theorem 13 for SpMvSVMs, we carried out experiments on the text classification data with a priori fixed regularization values $\gamma_n = 10^{-5}$ and $\gamma_v = 0.1$. This choice of parameters did not intend to be optimal in terms of test errors, but attempted to show the relationship, if any, between the generalization bound and the test error. The empirical Rademacher complexity



(a) Test accuracy (b) # Retained Unlabeled Examples

Figure 6: Classification accuracies and numbers of retained unlabeled examples for sequential training of SpMvSVMs. Parameter ε varies in $\{0.1, 0.2\}$.

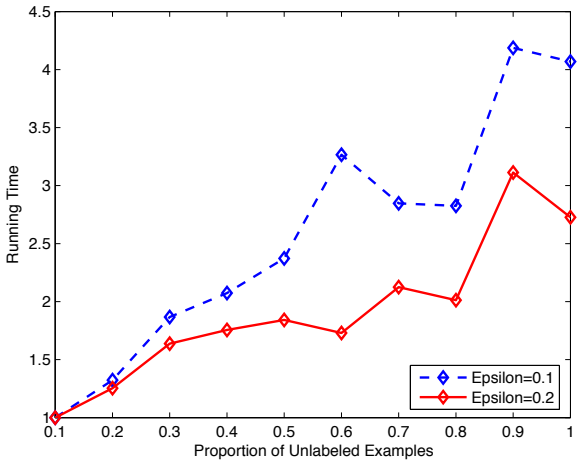
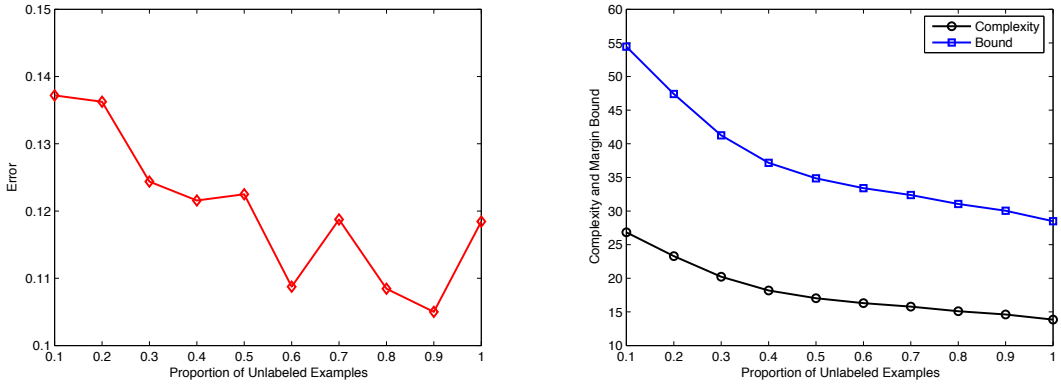


Figure 7: Running time for sequential training of SpMvSVMs. Parameter ε varies in $\{0.1, 0.2\}$. We have normalized the running times with 10% unlabeled examples to be 1.

$\hat{R}_\ell(\mathcal{G})$ in the margin bound is replaced by its upper bound $\mathcal{U}/\ell$ with $\mathcal{U}^2 = tr(\mathcal{S}) - \gamma_v tr(\mathcal{J}^\top (I + \gamma_v \Theta)^{-1} \mathcal{J})$.

For SpMvSVMs, only one iteration was performed in order to apply the margin bound. In other words, we learned classifiers only with the initially provided conjugate vector $\mathbf{z}$ whose entries were fixed as 1 for labeled data and 0.995 for unlabeled data. We used the same data split as in Section 6.2, but varied the size of unlabeled training set from $\{10\%, 20\%, \ldots, 100\%\}$ of all the available unlabeled data. To compute the margin bound, the confidence level in Theorem 13 is fixed as 95% ($\delta = 0.05$). The test error rate, empirical Rademacher complexity, and margin bound are



(a) Test error rate

(b) Complexity and margin bound

Figure 8: Classification error rates, empirical Rademacher complexity, and the margin bound of SpMvSVMs with different sizes of unlabeled sets.

shown in Figure 8. The overall decrease of error rates is well explained by the drop of the margin bound and empirical Rademacher complexity brought by the regularization role of more and more unlabeled data. Figure 8(b) also indicates that after adding a certain number of unlabeled data, including more unlabeled data will only improve the performance marginally. This phenomenon is observed in Figure 8(a) as well.

7. Extensions

In this section, we discuss possible extensions of using conjugate functions for sparse semi-supervised learning. In particular, the ε-insensitive loss term can be replaced by a somewhat general convex function. We also briefly discuss two sparse variants for Co-RLS and Laplacian SVMs using the same approach as sparse multi-view SVMs.

7.1 Arbitrary Convex Loss

In Section 2.2, for each example the ε-insensitive loss used is $f_\varepsilon(t_i) = (\sqrt{t_i} - \varepsilon)_+^2$ with $t_i = [f_1(x_i) - f_2(x_i)]^2$. This can be relaxed to a general class of user-designed losses that can be defined as a convex function of t_i, for example, using existing convex functions or compositions of convex functions with some good properties (Boyd and Vandenberghe, 2004).

Provided the ε-insensitive loss conforms the slight assumptions closed, convex, and proper listed in Lemma 5, the methodology used for sparse multi-view SVMs and the advantages of using Fenchel-Legendre conjugates apply well to the new optimization problem. This is an important contribution of this paper, which gives a framework for solving problems involving different ε-insensitive loss functions. Also, this framework applies to problems with a single view or more than two views, as long as the objective function is convex with respect to θ (parameters of classifiers or regressors) as in (11).

7.2 A Sparse Variant for Co-RLS

The objective function of Co-RLS in the case of two views is given as follows (Sindhwani et al., 2005; Brefeld et al., 2006)

$$\min_{f_1\in\mathcal{H}_1, f_2\in\mathcal{H}_2} \quad \frac{1}{2\ell}\sum_{i=1}^{\ell}[(f_1(x_i)-y_i)^2+(f_2(x_i)-y_i)^2]+ \\ \gamma_n(\|f_1\|^2+\|f_2\|^2)+\gamma_v\sum_{i=1}^{\ell+u}(f_1(x_i)-f_2(x_i))^2,$$

where nonnegative scalars γ_n, γ_v are respectively norm regularization and multi-view regularization coefficients. This optimization problem is indeed convex with respect to expansion coefficients α_1 and α_2 which have the same meanings as in (9).

Replacing the last term with $\sum_{i=1}^{\ell+u}(|f_1(x_i)-f_2(x_i)|-\varepsilon)_+^2$ results in the sparse Co-RLS algorithm

$$\min_{f_1\in\mathcal{H}_1, f_2\in\mathcal{H}_2} \quad \frac{1}{2\ell}\sum_{i=1}^{\ell}[(f_1(x_i)-y_i)^2+(f_2(x_i)-y_i)^2]+ \\ \gamma_n(\|f_1\|^2+\|f_2\|^2)+\gamma_v\sum_{i=1}^{\ell+u}(|f_1(x_i)-f_2(x_i)|-\varepsilon)_+^2.$$

This ε-insensitive loss is identical to that used in sparse multi-view SVMs, and therefore we can directly use the technique developed in this paper to solve this optimization.

7.3 A Sparse Variant for Laplacian SVMs

By including a penalty term on the intrinsic manifold smoothness, Belkin et al. (2006) proposed the Laplacian SVMs as an extension of SVMs by solving the following problem in an RKHS

$$\min_{f\in\mathcal{H}}\frac{1}{\ell}\sum_{i=1}^{\ell}(1-y_i f(x_i))_+ +\gamma_A\|f\|^2+\gamma_I\sum_{i,j=1}^{\ell+u}W_{ij}(f(x_i)-f(x_j))^2, \tag{33}$$

where $\mathcal{H}$ is the RKHS induced by a kernel, γ_A and γ_I are respectively ambient and intrinsic regularization coefficients, and $W_{ij}\geq 0$ are entries of the weight matrix W of the graph representing the manifold. The last term can be rewritten as

$$\begin{aligned}\sum_{i,j=1}^{\ell+u}W_{ij}(f(x_i)-f(x_j))^2 &= 2\left[\sum_{i=1}^{\ell+u}(\sum_{j=1}^{\ell+u}W_{ij})f^2(x_i)-\sum_{i,j=1}^{\ell+u}W_{ij}f(x_i)f(x_j)\right]\\ &= 2\mathbf{f}^\top(V-W)\mathbf{f}=2\mathbf{f}^\top\mathcal{L}\mathbf{f},\end{aligned}$$

where $\mathbf{f}=[f(x_1),\ldots,f(x_{\ell+u})]^\top$, matrix V is diagonal with the ith diagonal entry $V_{ii}=\sum_{j=1}^{\ell+u}W_{ij}$, and $\mathcal{L}$ is the positive semi-definite graph Laplacian.

This optimization problem is also convex with respect to expansion coefficient α, slack variable ξ and bias b if we formulate it as in (12). Replacing the last term in (33) with $\sum_{i,j=1}^{\ell+u}W_{ij}(|f(x_i)-f(x_j)|-\varepsilon)_+^2$ results in the sparse Laplacian SVMs

$$\min_{f\in\mathcal{H}}\frac{1}{\ell}\sum_{i=1}^{\ell}(1-y_i f(x_i))_+ +\gamma_A\|f\|^2+\gamma_I\sum_{i,j=1}^{\ell+u}W_{ij}(|f(x_i)-f(x_j)|-\varepsilon)_+^2,$$

This ε-insensitive loss has a similar form with that used in sparse multi-view SVMs, and thus facilitates an extension of the technique developed in this paper to solve this optimization.

8. Conclusion

In this paper, we proposed a sparse semi-supervised learning framework using Fenchel-Legendre conjugates. It is extendable to a wide range of semi-supervised learning methods. In particular, we formulated and solved the sparse multi-view SVMs, which incorporate an ε-insensitive multi-view regularization term. By rewriting this regularization in terms of conjugate functions, we obtained an inf-sup optimization problem whose globally optimal solutions can be found by our proposed iterative algorithm. We also showed that the quadratic program involved in each iteration only depends on the size of the labeled set, which would be very efficient for semi-supervised learning problems. For sparse multi-view SVMs, we characterized their generalization error in terms of the margin bound and derived the empirical Rademacher complexity of the considered function class. The empirical Rademacher complexity has two different forms depending on whether the iterative algorithm iterates only once or multiple steps.

Experimental results on sparse multi-view SVMs with different ε values showed that it is unnecessary to retain all the unlabeled data to represent target functions and using sparse semi-supervised learning can effectively reach a good balance between classifier performance and the number of unlabeled examples retained. This would be beneficial to speed up function evaluations during the classification of new examples. Comparisons with SVM-2K showed the superiority of our proposed method both on classification accuracies and the possibility and potential to be applied to large-scale problems when a sequential training strategy is adopted. As in this paper we only concern the possibility and potential for large-scale applications, we employed a moderate data set. It leaves as future work to apply the approach to much larger data sets. We also performed experiments to validate the usefulness of the margin bound and empirical Rademacher complexity, which explain well the regularization role unlabeled data play for multi-view learning.

Acknowledgments

We would like to thank Dr. Sandor Szedmak for providing the code of SVM-2K and related discussions. This work is supported in part by the National Natural Science Foundation of China under Project 61075005, Shanghai Educational Development Foundation under Project 2007CG30, and the PASCAL network of excellence.

References

M. Balcan and A. Blum. A PAC-style model for learning from labeled and unlabeled data. In *Proceedings of the 18th Annual Conference on Computational Learning Theory*, pages 111–126, 2005.

M. Balcan, A. Blum, and K. Yang. Co-training and expansion: Towards bridging theory and practice. *Advances in Neural Information Processing Systems*, 17:89–96, 2005.

P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled exampls. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, England, 2004.

U. Brefeld, T. Gärtner, T. Sheffer, and S. Wrobel. Efficient co-regularized least squares regression. In *Proceedings of the 23th International Conference on Machine Learning*, pages 137–144, 2006.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised Learning*. MIT Press, Cambridge, MA, 2006.

J. Farquhar, D. Hardoon, H. Meng, J. Shawe-Taylor, and S. Szedmak. Two view learning: SVM-2K, theory and practice. *Advances in Neural Information Processing Systems*, 18:355–362, 2006.

G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, USA, 1996.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

R. Latala and K. Oleszkiewicz. On the best constant in the Khintchine-Kahane inequality. *Studia Mathematica*, 109(1):101–104, 1994.

K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 9th International Conference on Information and Knowledge Management*, pages 86–93, 2000.

M. F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

R. Rifkin and R. Lippert. Value regularization and Fenchel duality. *Journal of Machine Learning Research*, 8:441–479, 2007.

D. Rosenberg and P. Bartlett. The Rademacher complexity of co-regularized kernel classes. *Journal of Machine Learning Research Workshop and Conference Proceedings*, 2:396–403, 2007.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, England, 2004.

V. Sindhwani and D. Rosenberg. An RKHS for multi-view learning and manifold co-regularization. In *Proceedings of the 25th International Conference on Machine Learning*, pages 976–983, 2008.

V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of the Workshop on Learning with Multiple Views, 22nd ICML*, 2005.

S. Sun. Semantic features for multi-view semi-supervised and active learning of text classification. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, pages 731–735, 2008.

S. Szedmak and J. Shawe-Taylor. Synthesis of maximum margin and multiview learning using unlabeled data. *Neurocomputing*, 70(7-9):1254–1264, 2007.

I. Tsang and J. Kwok. Large-scale sparsified manifold regularization. *Advances in Neural Information Processing Systems*, 19:1401–1408, 2007.

X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin Madison, 2008.

Journal of Machine Learning Research 11 (2010) 2457-2485 Submitted 9/09; Revised 5/10; Published 9/10

Rademacher Complexities and Bounding the Excess Risk in Active Learning

Vladimir Koltchinskii[*] VLAD@MATH.GATECH.EDU
School of Mathematics
Georgia Institute of Technology
Atlanta, GA 30332-0160, USA

Editor: Gabor Lugosi

Abstract

Sequential algorithms of active learning based on the estimation of the level sets of the empirical risk are discussed in the paper. Localized Rademacher complexities are used in the algorithms to estimate the sample sizes needed to achieve the required accuracy of learning in an adaptive way. Probabilistic bounds on the number of active examples have been proved and several applications to binary classification problems are considered.

Keywords: active learning, excess risk, Rademacher complexities, capacity function, disagreement coefficient

1. Introduction

Let $(S,\mathcal{A})$ be a measurable space and $T \subset \mathbb{R}$. Consider a standard **prediction problem** in which (X,Y) is a random couple in $S \times T$ with unknown distribution P. Here X is a **design point** whose distribution will be denoted Π and Y is a **response variable** with conditional distribution (given X) $P_{Y|X}(\cdot|X=x)$. The response variable Y is to be predicted based on an observation of X. This class of problems includes many versions of regression and classification. For instance, in the binary classification, $T=\{-1,1\}$ and the conditional distribution of Y given X is completely characterized by the regression function

$$\eta(x) := \mathbb{E}(Y|X=x).$$

In the framework of **passive learning**, a learning algorithm inputs the training data $(X_1,Y_1),\dots (X_n,Y_n)$ that consists of n independent examples sampled from the distribution P. The goal is to construct a data dependent prediction rule $\hat{g}: S \mapsto T$ whose risk with respect to a properly chosen loss function is "close" to the minimal possible risk. More precisely, given a loss function $\ell : T \times T \mapsto \mathbb{R}$, the risk of a prediction rule $g : S \mapsto T$ is defined as

$$P(\ell \bullet g) = \mathbb{E}\ell(Y; g(X)),$$

where we used the notation $(\ell \bullet g)(x,y) := \ell(y; g(x))$. For instance, in the binary classification setting, the binary loss $\ell(y,u) := I(y \neq u)$ is usually used. In this case, the risk of a classification rule $g : S \mapsto \{-1,1\}$ is its generalization error:

$$P(\ell \bullet g) = \mathbb{P}\{Y \neq g(X)\}.$$

∗. Partially supported by NSF grants MSPA-MCS-0624841, DMS-0906880 and CCF-0808863.

Suppose $\mathcal{G}$ is a class of prediction rules $g : S \mapsto T$. The quantity

$$\mathcal{E}_P(\ell \bullet g) := P(\ell \bullet g) - \inf_{g \in \mathcal{G}} P(\ell \bullet g)$$

is called the **excess risk** of g. One of the main goals of statistical analysis of learning algorithms is to understand how the excess risk $\mathcal{E}_P(\ell \bullet \hat{g})$ of a data dependent decision rule $\hat{g}$ output by such an algorithm depends on the sample size n, on the "complexity" of the class $\mathcal{G}$ of prediction rules and on the underlying complexity of the prediction problem itself.

In the recent years, there has been a lot of interest in **active learning** algorithms. In this framework, the algorithm can modify the design distribution in the process of learning. More precisely, suppose that the training examples (X_j, Y_j) are sampled sequentially. At each iteration (say, iteration number k), the algorithm requests a design point X_{k+1} sampled from a distribution $\hat{\Pi}_k$ that depends on the training data $(X_1, Y_1), \ldots, (X_k, Y_k)$. Given $X_{k+1} = x$, the response variable (the label) Y_{k+1} is sampled from the conditional distribution $P_{Y|X}(\cdot|X = x)$. The question is whether there are such active learning algorithms for which the excess risk after n iterations is provably smaller than for the passive prediction rules based on the same number n of training examples. It happens that the answer depends on the type of learning problem. A minimax analysis by Castro et al. (2005) and Castro and Nowak (2008) shows that such an improvement is possible in classification problems and in some special classes of regression problems with non-smooth regression function (for instance, if the regression function is a step function). In such cases, the improvement can be very significant. In some classification problems, the excess risk of active learning algorithms can converge to zero with an **exponential rate** as $n \to \infty$ (comparing with the rate $O(n^{-1})$ in the case of passive learning). Castro and Nowak (2008) studied several examples of binary classification problems in which the active learning approach is beneficial and suggested nice active learning algorithms in these problems. However, the drawback of these algorithms is that they are not adaptive in the sense that they require the prior knowledge of distribution dependent parameters of the problem, such as noise characteristics in classification. The development of active learning methods that are adaptive and, at the same time, computationally tractable remains a challenge. There has been a progress in the design of active learning algorithms that possess some degree of adaptivity, in particular, see Dasgupta et al. (2007), Balcan et al. (2009), Balcan et al. (2008) and Hanneke (2009a,b). In the last two interesting papers by Hanneke, some versions of the algorithms of Balcan et al. (2009) and of Dasgupta et al. (2007) were studied using the technique of **Rademacher complexities** that much earlier proved to be very useful in the analysis of passive learning, see Bartlett et al. (2002), Koltchinskii (2001), Koltchinskii and Panchenko (2000), Bartlett et al. (2005), Koltchinskii (2006, 2008) and references therein. Hanneke showed that incorporating Rademacher complexities in active learning algorithms allows one to develop rather general versions of such algorithms that are adaptive under broad assumptions on the underlying distributions.

In the current paper, we continue this line of research. We consider the following model of active learning. At each iteration, a learning algorithm has to choose a set $\hat{A} \subset S$ of "good" design points and also the number of training examples needed at the current iteration. Both the set $\hat{A}$ and the required number of the examples might depend on the training data that is already available. The algorithm has an access to an oracle that is asked to provide the required number of examples (X, Y) sampled from the conditional distribution $P(\cdot|x \in \hat{A})$. Alternatively, it can be described as follows. The oracle provides training examples (X, Y) sampled from an unknown probability distribution P. At each iteration, the algorithm chooses a set $\hat{A}$ of "good" design points and asks the oracle

whether the next design point X_k is "good". If it is "good", the algorithm accepts the point, the oracle provides it with a label Y_k and returns the couple (X_k, Y_k). Testing whether the example is "good" costs the algorithm nothing, but each "good" labeled training example costs \$ 1. Thus, only the number of "good" examples matters for determining the total cost of learning. The question is how many "good" examples are needed for the excess risk $\mathcal{E}_P(\ell \bullet \hat{g})$ of the resulting classifier $\hat{g}$ to become smaller than δ with a guaranteed probability at least $1-\alpha$.

We will develop active learning algorithms that are somewhat akin to what is done by Hanneke (2009a,b), but they are more closely related to the construction of localized Rademacher complexities used in the definitions of distribution dependent and data dependent excess risk bounds in empirical risk minimization, see Koltchinskii (2006, 2008). The main idea of this construction is to characterize the complexity of the problem by the sup-norm of a special Rademacher process indexed by the level sets of the risk. To be more specific, suppose that we are dealing with a binary classification problem and that the empirical risk (with respect to the binary loss) is being minimized over a class $\mathcal{G}$ of binary functions. Then what matters is the collection of δ-minimal sets

$$\mathcal{G}(\delta) := \Big\{ g \in \mathcal{G} : \mathcal{E}_P(\ell \bullet g) \leq \delta \Big\}, \ \delta > 0.$$

These sets can be estimated based on the empirical data and Rademacher complexities of such estimated sets for small enough values of δ are used to define reasonably tight bounds on the excess risk. In many learning problems, the δ-minimal sets become small as $\delta \to 0$, for instance, in the sense that their $L_2(\Pi)$-diameter is small. It turns out that an important role in the development of active classification algorithms is played by the sets of the following type

$$A(\delta) := \Big\{ x \in S | \exists g_1, g_2 \in \mathcal{G}(\delta) : g_1(x) \neq g_2(x) \Big\}.$$

Such a set is called a **disagreement set** since it consists of the points for which there are two classifiers in $\mathcal{G}(\delta)$ whose predictions at point x disagree with each other. If the δ-minimal sets are small for small enough values of δ, one can expect that the corresponding disagreement sets are also small. This is not always the case, but there are natural examples in which indeed the measure $\Pi(A(\delta))$ tends to 0 as $\delta \to 0$ (sometimes, even $\Pi(A(\delta)) = O(\delta)$). Note that if the empirical risk is being minimized over the δ-minimal set $\mathcal{G}(\delta)$, one can eliminate from the sample all the design points X_j such that $X_j \notin A(\delta)$: the minimizers of the empirical risk are not going to change since the value of the binary loss at such training examples (X_j, Y_j) is a constant on the δ-minimal set. So, only the examples for which X_j belongs to a small disagreement set $A(\delta)$ are really needed. This simple observation opens a possibility of reducing the sample size in the process of active learning, and this has already been exploited in several algorithms described in the literature: see Dasgupta et al. (2007), Balcan et al. (2009), Hanneke (2009a,b) and references therein. It is interesting to mention that some notions similar to the "disagreement sets" were used much earlier in the study of ratio type empirical processes (for instance, in the work of Alexander in the 80s; see, Giné and Koltchinskii, 2006 and references therein). Moreover, it was used by Giné and Koltchinskii (2006) to obtain refined excess risk bounds in binary classification (in the passive learning case). This will be discussed in some detail in Section 4.

Our approach is based on iterative estimation of the δ-minimal sets for a decreasing sequence $\{\delta_j\}$ of values of δ. It happens that, for larger values of δ, it is possible to construct a rough estimate of the δ-minimal sets based on a relatively small number of training examples. The required

sample sizes $\bar{n}(\delta)$ can be estimated using Rademacher complexities. For smaller values of δ, more examples are needed, but, at the same time, for the smaller values of δ the disagreement sets are also small, and these sets again can be estimated based on the training examples that have been already sampled. Thus, there is a possibility to come up with an active learning strategy that, at each iteration, computes an estimate $\hat{A}$ of the disagreement set and determines the required sample size, and then samples the required number of design points from the conditional distribution $\Pi(\cdot|x \in \hat{A})$. Each of these points $X_j = x$ is provided with a label Y_j sampled from the conditional distribution $P_{Y|X}(\cdot|X = x)$. The algorithm stops as soon as δ_j becomes smaller than the required accuracy of learning δ. At this stage, it outputs an estimate of the δ-minimal set $\mathcal{G}(\delta)$. The number of labeled examples needed to achieve this goal is roughly

$$\sum_{\delta_j \geq \delta} \bar{n}(\delta_j)\Pi(A(\delta_j))$$

(see theorems 7, 8, 9 for more precise formulations). For binary classification problems with a VC-class $\mathcal{G}$ such that $\Pi(A(\delta)) = O(\delta)$, this leads to the bound on the number of labeled examples of the order $\log(1/\delta)\log\log(1/\delta)$.

The need to estimate the whole δ-minimal set in this learning strategy rather than simply minimizing the empirical risk might look like too strong of an assumption. However, in the alternative general versions of adaptive strategies of active learning due to Hanneke (2009a,b) this is also needed. Hanneke uses previously suggested agnostic learning methods of Dasgupta et al. (2007) and of Balcan et al. (2009) in combination with Rademacher complexities that are based on estimated level sets of the empirical risk. So, currently, this seems to be unavoidable in general adaptive methods and our approach is just based on a more direct use of the δ-minimal sets.

To give a precise description of a version of active learning method considered in this paper and to study its statistical properties, several facts from the general theory of empirical risk minimization will be needed. In particular, in Section 2, we describe a construction of distribution dependent and data dependent bounds on the excess risk based on localized Rademacher complexities, see Koltchinskii (2006, 2008). In Section 3, we describe our active learning algorithms. These algorithms are sequential in the sense that the training data is being sampled until the desired accuracy of learning is achieved. We prove several bounds on the number of active examples needed to achieve this goal with a specified probability. In sections 2 and 3, it is convenient to study the problem in a more abstract framework, in which we suppress the labels Y_j and write S instead of $S \times \{-1,1\}$, X instead of (X,Y), f instead of $\ell \bullet g$, etc. This allows us to simplify the notations and the description of the algorithms, and, at the same time, it makes the results a little more general. In principle, it should be possible to apply these results to more general classes of learning problems than binary classification (for instance, to special regression models with a non-smooth regression function or to the problem of estimation of the level sets of an unknown probability density). However, we do not pursue this possibility here and, instead, we concentrate in Section 4 on the binary classification problems, which still remains the most interesting class of learning problems where the active learning approach leads to faster convergence rates.

2. Empirical Risk Minimization: Bounds on the Excess Risk

Let $(S,\mathcal{A})$ be a measurable space, let P be a probability measure in $(S,\mathcal{A})$ and let $X, X_1, X_2, \ldots$ be i.i.d. random variables in $(S,\mathcal{A})$ with common distribution P. Let $\mathcal{F}$ be a class of $\mathcal{A}$-measurable

functions $f : S \mapsto [0,1]$. The values of functions $f \in \mathcal{F}$ will be interpreted as "losses" associated with some decisions and the integral

$$Pf := \int_S f dP = \mathbb{E} f(X)$$

represents the expected loss, or the (true) risk. The optimization problem

$$Pf \longrightarrow \min,\ f \in \mathcal{F}$$

is interpreted as a "risk minimization" problem and the quantity

$$\mathcal{E}_P(f) := Pf - \inf_{g \in \mathcal{F}} Pg$$

is called the excess risk of f. The δ-minimal set of the true risk is defined as

$$\mathcal{F}_P(\delta) := \left\{ f \in \mathcal{F} : \mathcal{E}_P(f) \le \delta \right\},\ \delta \ge 0.$$

In learning theory problems, the distribution P is usually unknown and the risk Pf is to be estimated by the empirical risk. The empirical measure based on the sample $(X_1, \dots, X_n)$ of size n is defined as

$$P_n := n^{-1} \sum_{j=1}^{n} \delta_{X_j}$$

and the problem of risk minimization is replaced by the "empirical risk minimization":

$$P_n f \longrightarrow \min,\ f \in \mathcal{F}. \tag{1}$$

Naturally, this also leads to the definitions of the "excess empirical risk" $\mathcal{E}_{P_n}(f)$ and of the δ-minimal sets of the empirical risk $\mathcal{F}_{P_n}(\delta),\ \delta > 0$.

Given a solution $\hat{f}$ of the empirical risk minimization problem (1), a basic question is to provide reasonably tight upper confidence bounds on the excess risk $\mathcal{E}_P(\hat{f})$ that depend on complexity characteristics of the class $\mathcal{F}$. It is also of importance to understand when the δ-minimal sets of the empirical risk are reasonably good estimates of the δ-minimal sets of the true risk. We will need below several results of this type that can be found in the papers by Koltchinskii (2006, 2008).

First of all, we will need an upper confidence bound on the size of the empirical process

$$\sup_{f,g \in \mathcal{F}_P(\delta)} |(P_n - P)(f-g)|.$$

To construct such a bound, we use Talagrand's famous concentration inequalities. Suppose $\rho_P : L_2(P) \times L_2(P) \mapsto [0, +\infty)$ and

$$\rho_P^2(f,g) \ge P(f-g)^2 - (P(f-g))^2,\ f,g \in L_2(P).$$

Define the diameter of $\mathcal{F}_P(\delta)$ as

$$D(\delta) := D_P(\delta) := \sup_{f,g \in \mathcal{F}(\delta)} \rho_P(f,g).$$

It provides a measure of the size of the δ-minimal sets. We will also use the following quantity that characterizes the accuracy of "empirical approximation" of P by P_n on the δ-minimal sets:

$$\phi_n(\delta) := \mathbb{E} \sup_{f,g\in\mathcal{F}(\delta)} |(P_n - P)(f-g)|.$$

Given a decreasing sequence $\{\delta_j\}$ of positive numbers with $\delta_0 := 1$ and a sequence $\{t_j\}$ of positive numbers, define a step function $U_n(\delta),\ \delta \in (0,1]$ as follows:

$$\bar{U}_n(\delta) := 2\sum_{j\geq 0}\left[\phi_n(\delta_j) + D(\delta_j)\sqrt{\frac{t_j}{n}} + \frac{t_j}{n}\right] I_{(\delta_{j+1},\delta_j]}(\delta).$$

A version of Talagrand's concentration inequality with explicit constants due to Bousquet implies that, for all $j \geq 0$ and for all $\delta \in (\delta_{j+1},\delta_j]$, with probability at least $1 - e^{-t_j}$

$$\sup_{f,g\in\mathcal{F}_P(\delta)} |(P_n - P)(f-g)| \leq \bar{U}_n(\delta).$$

Given $\psi : \mathbb{R}_+ \mapsto \mathbb{R}_+$, define

$$\psi^\flat(\delta) := \sup_{\sigma\geq\delta} \frac{\psi(\sigma)}{\sigma}$$

and

$$\psi^\sharp(\varepsilon) := \inf\left\{\delta > 0 : \psi^\flat(\delta) \leq \varepsilon\right\}.$$

Let

$$\delta_n(\mathcal{F};P) := \sup\left\{\delta \in (0,1] : \delta \leq \bar{U}_n(\delta)\right\}.$$

The following bounds were proved by Koltchinskii (2006, 2008).

Theorem 1 *For all* $\delta \geq \delta_n(\mathcal{F};P)$,

$$\mathbb{P}\left\{\mathcal{E}_P(\hat{f}) > \delta\right\} \leq \sum_{\delta_j\geq\delta} e^{-t_j}$$

and

$$\mathbb{P}\left\{\sup_{f\in\mathcal{F},\mathcal{E}_P(f)\geq\delta}\left|\frac{\mathcal{E}_{P_n}(f)}{\mathcal{E}_P(f)} - 1\right| > \bar{U}_n^\flat(\delta)\right\} \leq \sum_{\delta_j\geq\delta} e^{-t_j}.$$

Thus, the quantity $\delta_n(\mathcal{F};P)$ is a distribution dependent upper bound on the excess risk $\mathcal{E}_P(\hat{f})$ that holds with a guaranteed probability. Moreover, for all $\delta \geq \delta_n(\mathcal{F};P)$ and for all $f \in \mathcal{F}$ with $\mathcal{E}_P(f) \geq \delta$ it is possible to control the size of the ratio $\frac{\mathcal{E}_{P_n}(f)}{\mathcal{E}_P(f)}$ in terms of the quantity $\bar{U}_n^\flat(\delta)$. This ratio bound for the excess risk immediately implies the following statement showing that for all the values of δ above certain threshold the δ-minimal sets of empirical risk provide estimates of the δ-minimal sets of the true risk.

Proposition 2 *Let* $\bar{\delta}_n := \bar{U}_n^{\sharp}\left(\frac{1}{2}\right)$. *For all* $\delta \geq \bar{\delta}_n$, *with probability at least*

$$1 - \sum_{\delta_j \geq \delta} e^{-t_j}$$

the following inclusions hold:

$$\forall \sigma \geq \delta \;\; \mathcal{F}_P(\sigma) \subset \mathcal{F}_{P_n}(3/2\sigma) \;\text{ and }\; \mathcal{F}_{P_n}(\sigma) \subset \mathcal{F}_P(2\sigma).$$

Data dependent upper confidence bounds on the excess risk can be constructed using localized sup-norms of Rademacher processes that provide a way to estimate the size of the empirical process. Given i.i.d. Rademacher random variables $\{\varepsilon_i\}$ independent of $\{X_i\}$, the Rademacher process is defined as

$$R_n(f) := n^{-1} \sum_{j=1}^{n} \varepsilon_j f(X_j).$$

We will assume that

$$\rho_P^2(f,g) := P(f-g)^2.$$

Define

$$\hat{\phi}_n(\delta) := \sup_{f,g \in \mathcal{F}_{P_n}(\delta)} |R_n(f-g)|$$

and

$$\hat{D}_n(\delta) := \sup_{f,g \in \mathcal{F}_{P_n}(\delta)} \rho_{P_n}(f,g).$$

These quantities are empirical versions of $\phi_n(\delta)$ and $D_P(\delta)$ and they can be used to define an empirical version of the function $\bar{U}_n$:

$$\hat{U}_n(\delta) := \hat{K} \sum_{j \geq 0} \left[\hat{\phi}_n(\hat{c}\delta_j) + \hat{D}_n(\hat{c}\delta_j)\sqrt{\frac{t_j}{n}} + \frac{t_j}{n} \right] I_{(\delta_{j+1},\delta_j]}(\delta),$$

where $\hat{K}, \hat{c}$ are numerical constants. We will also define

$$\tilde{U}_n(\delta) := \tilde{K} \sum_{j \geq 0} \left[\phi_n(\tilde{c}\delta_j) + D(\tilde{c}\delta_j)\sqrt{\frac{t_j}{n}} + \frac{t_j}{n} \right] I_{(\delta_{j+1},\delta_j]}(\delta)$$

with some numerical constants $\tilde{K}, \tilde{c}$.

It can be shown (see Koltchinskii, 2006, 2008) that for large enough numerical constants $\hat{K}, \tilde{K}, \hat{c}, \tilde{c}$ and for all $\delta \geq \bar{\delta}_n$, $\bar{U}_n(\delta) \leq \hat{U}_n(\delta) \leq \tilde{U}_n(\delta)$ with a high probability. More precisely, the following statement holds. Denote

$$\hat{\delta}_n := \hat{U}_n^{\sharp}\left(\frac{1}{2}\right),\ \tilde{\delta}_n := \tilde{U}_n^{\sharp}\left(\frac{1}{2}\right).$$

Theorem 3 *There exists a choice of numerical constants* $\hat{K}, \tilde{K}, \hat{c}, \tilde{c}$ *in the definitions of the functions* $\hat{U}_n, \tilde{U}_n$ *such that the following holds. For all* $\delta \geq \bar{\delta}_n$, *there exists an event* E *of probability*

$$\mathbb{P}(E) \geq 1 - 3 \sum_{\delta_j \geq \delta} e^{-t_j}$$

such that on this event

$$\bar{U}_n(\sigma) \le \hat{U}_n(\sigma) \le \tilde{U}_n(\sigma),\ \sigma \ge \delta.$$

As a consequence,

$$\mathbb{P}\left\{\bar{\delta}_n \le \hat{\delta}_n \le \tilde{\delta}_n\right\} \ge 1 - 3\sum_{\delta_j \ge \bar{\delta}_n} e^{-t_j}.$$

The proof is based on the following "statistical version" of Talagrand's concentration inequality (which, in turn, follows from the usual Talagrand's concentration inequality for empirical processes and standard symmetrization and contraction arguments, see Koltchinskii, 2008). Suppose that $\mathcal{F}$ is a class of measurable functions on S uniformly bounded by $U > 0$. Denote

$$\sigma_P^2(\mathcal{F}) := \sup_{f\in\mathcal{F}} Pf^2 \text{ and } \sigma_n^2(\mathcal{F}) := \sup_{f\in\mathcal{F}} P_n f^2.$$

For a function $Y : \mathcal{F} \mapsto \mathbb{R}$, denote

$$\|Y\|_{\mathcal{F}} := \sup_{f\in\mathcal{F}} |Y(f)|.$$

Theorem 4 *There exists a numerical constant $K > 0$ such that for all $t \ge 1$ with probability at least $1 - e^{-t}$ the following bounds hold:*

$$\left|\|R_n\|_{\mathcal{F}} - \mathbb{E}\|R_n\|_{\mathcal{F}}\right| \le K\left[\sqrt{\frac{t}{n}\left(\sigma_n^2(\mathcal{F}) + U\|R_n\|_{\mathcal{F}}\right)} + \frac{tU}{n}\right],$$

$$\mathbb{E}\|R_n\|_{\mathcal{F}} \le K\left[\|R_n\|_{\mathcal{F}} + \sigma_n(\mathcal{F})\sqrt{\frac{t}{n}} + \frac{tU}{n}\right],$$

$$\sigma_P^2(\mathcal{F}) \le K\left(\sigma_n^2(\mathcal{F}) + U\|R_n\|_{\mathcal{F}} + \frac{tU}{n}\right)$$

and

$$\sigma_n^2(\mathcal{F}) \le K\left(\sigma_P^2(\mathcal{F}) + U\|R_n\|_{\mathcal{F}} + \frac{tU}{n}\right).$$

Also,

$$\mathbb{E}\|P_n - P\|_{\mathcal{F}} \le K\left[\|R_n\|_{\mathcal{F}} + \sigma_n(\mathcal{F})\sqrt{\frac{t}{n}} + \frac{tU}{n}\right]$$

and

$$\left|\|P_n - P\|_{\mathcal{F}} - \mathbb{E}\|P_n - P\|_{\mathcal{F}}\right| \le K\left[\sqrt{\frac{t}{n}\left(\sigma_n^2(\mathcal{F}) + U\|R_n\|_{\mathcal{F}}\right)} + \frac{tU}{n}\right].$$

In what follows, it will be of interest to consider sequential learning algorithms in which the sample size is being gradually increased until the excess risk becomes smaller than a given level δ. The following quantities are used in the analysis of such algorithms. Let us fix a set $M \subset \mathbb{N}$. A possible choice is $M = \mathbb{N}$, but, usually, we will take $M = \{2^k : k \ge 0\}$. Denote

$$\bar{n}(\delta) := \inf\left\{n \in M : \bar{\delta}_n \le \delta\right\} = \inf\left\{n \in M : \bar{U}_n^{\flat}(\delta) \le \frac{1}{2}\right\},$$

$$\hat{n}(\delta) := \inf\left\{ n \in M : \hat{\delta}_n \le \delta \right\} = \inf\left\{ n \in M : \hat{U}_n^{\flat}(\delta) \le \frac{1}{2} \right\}$$

and

$$\tilde{n}(\delta) := \inf\left\{ n \in M : \tilde{\delta}_n \le \delta \right\} = \inf\left\{ n \in M : \tilde{U}_n^{\flat}(\delta) \le \frac{1}{2} \right\}.$$

If

$$\mathbb{E}\|P_n - P\|_{\mathcal{F}} \to 0 \text{ as } n \to \infty,$$

which is true for so called Glivenko-Cantelli classes of functions with respect to P (see, e.g., van der Vaart and Wellner, 1996), then it is easy to see that

$$\bar{\delta}_n \to 0 \text{ and } \tilde{\delta}_n \to 0 \text{ as } n \to \infty.$$

In this case, we have

$$\tilde{n}(\delta) < +\infty, \bar{n}(\delta) < +\infty, \delta \in (0,1].$$

It is also easy to see that the functions $n \mapsto \bar{U}_n(\delta)$ and $n \mapsto \tilde{U}_n(\delta)$ are nonincreasing (it follows from the well known reverse supermartingale properties of empirical processes; see, van der Vaart and Wellner, 1996, Lemma 2.4.5). This implies that, for all $n \ge \bar{n}(\delta)$, $\bar{\delta}_n \le \delta$ and, for all $n \ge \tilde{n}(\delta)$, $\tilde{\delta}_n \le \delta$. Since $\bar{U}_n(\delta) \le \tilde{U}_n(\delta),\ \delta \in (0,1]$, it is also clear that

$$\bar{n}(\delta) \le \tilde{n}(\delta),\ \delta \in (0,1].$$

The next proposition immediately follows from the definition of $\bar{n}(\delta)$ (it is, in fact, just a reformulation of the statements of Proposition 2 and Theorem 3):

Proposition 5 *(i) For all $n \ge \bar{n}(\delta)$,*

$$\mathbb{P}\left\{ \mathcal{E}_P(\hat{f}_n) > \delta \right\} \le \sum_{\delta_j \ge \delta} e^{-t_j};$$

(ii) For all $n \ge \bar{n}(\delta)$, with probability at least

$$1 - \sum_{\delta_j \ge \delta} e^{-t_j}$$

the following inclusions hold:

$$\forall \sigma \ge \delta \ \ \mathcal{F}_P(\sigma) \subset \mathcal{F}_{P_n}(3/2\sigma) \ \text{ and } \ \mathcal{F}_{P_n}(\sigma) \subset \mathcal{F}_P(2\sigma).$$

(iii) For all $n \ge \bar{n}(\delta)$, there exists an event E of probability

$$\mathbb{P}(E) \ge 1 - 3\sum_{\delta_j \ge \delta} e^{-t_j}$$

such that on this event

$$\bar{U}_n(\sigma) \le \hat{U}_n(\sigma) \le \tilde{U}_n(\sigma),\ \sigma \ge \delta.$$

We will also need a version of the statements of Proposition 5 that are uniform in $n \in M$. To this end, assume that the numbers t_j in the definitions of the functions $\bar{U}_n, \hat{U}_n, \tilde{U}_n$ depend also on n. We will denote these numbers $t_j^{(n)}$. Assume that, for all j, $\frac{t_j^{(n)}}{n}$ is a nonincreasing function of n. Then the next statement immediately follows from Theorem 3 and the union bound.

Proposition 6 *There exists an event H of probability*

$$\mathbb{P}(H) \geq 1 - \sum_{n \in M} \sum_{j \geq 0} e^{-t_j^{(n)}}$$

such that on this event, for all $n \in M$,

$$\mathcal{E}_P(\hat{f}_n) \leq \bar{\delta}_n,$$

$$\forall \delta \geq \bar{\delta}_n \;\; \mathcal{F}_P(\delta) \subset \mathcal{F}_{P_n}(3/2\delta) \;\text{ and }\; \mathcal{F}_{P_n}(\delta) \subset \mathcal{F}_P(2\delta).$$

Moreover, there exists an event E of probability

$$\mathbb{P}(E) \geq 1 - 3 \sum_{n \in M} \sum_{j \geq 0} e^{-t_j^{(n)}}$$

such that on this event, for all $n \in M$,

$$\bar{U}_n(\delta) \leq \hat{U}_n(\delta) \leq \tilde{U}_n(\delta), \; \sigma \geq \bar{\delta}_n$$

and

$$\bar{\delta}_n \leq \hat{\delta}_n \leq \tilde{\delta}_n.$$

As a consequence, we also have that for all $\delta \in (0,1]$

$$\bar{n}(\delta) \leq \hat{n}(\delta) \leq \tilde{n}(\delta).$$

The simplest choice of the numbers $t_j^{(n)}$, in the case when $M = \{2^k : k \geq 0\}$ and $\delta_j = 2^{-j}, \; j \geq 0$, is

$$t_j^{(n)} = 2\log(\log_2 n + 1) + 2\log(j+1) + \log\frac{1}{\alpha} + \log(12), \; j \geq 0, \; n \in M,$$

where $\alpha \in (0,1)$. With this choice, all the claims of Proposition 6 hold with a guaranteed probability at least $1 - \alpha$.

The main conclusion one can draw from Proposition 6 is that the sample size needed to achieve the desired "accuracy of learning" δ (i.e., to "learn" a function for which the excess risk is smaller than δ) can itself be learned from the data. More precisely, the estimator $\hat{n}(\delta)$ of the required sample size can be computed sequentially by increasing the sample size n gradually, computing for each n the data dependent excess risk bound $\hat{\delta}_n$ and stopping as soon as $\hat{\delta}_n \leq \delta$. With a high probability, the stopping time $\hat{n}(\delta)$ provides a "correct" estimate of the sample size (up to a numerical constant) in the sense that it is between two distribution dependent estimates ($\bar{n}(\delta)$ and $\tilde{n}(\delta)$) that are typically of the same order of magnitude (up to numerical constants). At the same time, the sample size $\hat{n}(\delta)$ is sufficient for estimating the σ-minimal sets of the true risk by the σ-minimal sets of the empirical risk for all $\sigma \geq \delta$ (in the sense of the inclusions of Proposition 6). These facts will play a crucial role in our design of active learning methods in the next section.

3. Sequential Active Learning

We first describe a simplified (non-adaptive) version of active learning in which it is assumed that the minimal sample size $\bar{n}(\delta)$ needed to achieve the desired "accuracy of learning" of the order δ is given. As before, suppose that $\{\delta_k\}_{k\geq 0}$ is a nonincreasing sequence of positive numbers with $\delta_0 = 1$. Denote $\bar{n}_k := \bar{n}(\delta_k)$, $k \geq 1$.

Algorithm 1

$\hat{\mathcal{F}}_0 := \mathcal{F}$;
for $k = 1, 2, \ldots,$
$\hat{A}_k := \left\{x : \sup_{f,g\in\hat{\mathcal{F}}_{k-1}} |f(x) - g(x)| > \delta_k\right\}$;
$\hat{P}_k := \frac{1}{\bar{n}_k}\sum_{j=1}^{\bar{n}_k} I_{\hat{A}_k}(X_j)\delta_{X_j}$;
$\hat{\mathcal{F}}_k := \hat{\mathcal{F}}_{k-1}\bigcap \mathcal{F}_{\hat{P}_k}(3\delta_k)$;
end

The set $\hat{A}_k$ defined at each iteration of the algorithm is viewed as a set of "active examples" (or "active set"). The examples $X_j \in \hat{A}_k$ are needed to compute the "active empirical measure" $\hat{P}_k$. The underlying assumption is that there exists a "base algorithm" that computes the δ-minimal set

$$\mathcal{F}_Q(\delta) := \left\{f : \mathcal{E}_Q(f) := Qf - \inf_{f\in\mathcal{F}} Qf \leq \delta\right\}$$

for an arbitrary discrete measure Q with a finite number of atoms. This algorithm is used to compute the set $\mathcal{F}_{\hat{P}_k}(3\delta_k)$. In principle, it would be enough only to ensure that, given $\delta > 0$ and measure Q, the "base algorithm" outputs a set $\bar{\mathcal{F}}_Q(\delta)$ such that

$$\mathcal{F}_Q(c_1\delta) \subset \bar{\mathcal{F}}_Q(\delta) \subset \mathcal{F}_Q(c_2\delta)$$

for some numerical constants $0 < c_1 < c_2$. However, to simplify the notations, we will assume that $c_1 = c_2 = 1$.

Of course, in reality, **Algorithm 1** stops after a finite number of iterations. A possible choice could be

$$L := \sum_{j\geq 0} I(\delta_j \geq \delta),$$

which can be viewed as the number of iterations needed to achieve the "desired accuracy" of learning $\delta \in (0,1)$. In other words, the algorithm stops when δ_j becomes smaller than δ. For instance, if $\delta_j = 2^{-j}$, $j \geq 0$, then the number of iterations L is of the order $\log_2(1/\delta)$. Let $\nu(\delta)$ denote the total number of active examples used by the algorithm in the first L iterations. Then

$$\nu(\delta) \leq \sum_{\delta_k\geq\delta}\sum_{j=1}^{\bar{n}_k} I_{\hat{A}_k}(X_j).$$

Denote

$$A(\delta) := \left\{x : \sup_{f,g\in\mathcal{F}(8\delta)} |f(x) - g(x)| > \delta\right\}$$

and

$$\pi(\delta) := P(A(\delta)).$$

The following statement will be easily proved by induction.

Theorem 7 *With probability at least*

$$1-\sum_{n\in M}\sum_{j\geq 0}e^{-t_j^{(n)}},$$

the following inclusions hold for the classes $\hat{\mathcal{F}}_k$ *output by* **Algorithm 1***: for all* $k\geq 0$

$$\mathcal{F}_P(\delta_k)\subset\hat{\mathcal{F}}_k\subset\mathcal{F}_P(8\delta_k). \tag{2}$$

Also, for all $t\geq 1$ *and all* $\delta\in(0,1]$, *with probability at least*

$$1-\sum_{n\in M}\sum_{j\geq 0}\exp\{-t_j^{(n)}\}-\sum_{\delta_j\geq\delta}\exp\{-\bar{n}(\delta_j)\pi(\delta_{j-1})t\log t\}$$

the following bound holds:

$$\nu(\delta)\leq et\sum_{\delta_j\geq\delta}\bar{n}(\delta_j)\pi(\delta_{j-1}).$$

Proof The inclusions (2) obviously hold for $k=0$. Assuming that, for all $j<k$,

$$\mathcal{F}_P(\delta_j)\subset\hat{\mathcal{F}}_j\subset\mathcal{F}_P(8\delta_j),$$

we will prove that the same inclusions hold also for k. Let H be the event of probability at least

$$1-\sum_{n\in M}\sum_{j\geq 0}e^{-t_j^{(n)}}$$

defined in Proposition 6. By the induction assumption,

$$\mathcal{F}_P(\delta_k)\subset\mathcal{F}_P(\delta_{k-1})\subset\hat{\mathcal{F}}_{k-1}$$

and, by the definition of $\hat{A}_k$, we have for all $f,g\in\hat{\mathcal{F}}_{k-1}$,

$$|P_{\bar{n}_k}(f-g)-\hat{P}_k(f-g)|=\left|\bar{n}_k^{-1}\sum_{i=1}^{\bar{n}_k}(f-g)(X_i)-\bar{n}_k^{-1}\sum_{i:X_i\in\hat{A}_k}(f-g)(X_i)\right|$$

$$=\left|\bar{n}_k^{-1}\sum_{i:X_i\notin\hat{A}_k}(f-g)(X_i)\right|\leq\delta_k.$$

We can conclude that, for all $f\in\mathcal{F}_P(\delta_k)$,

$$\left|\mathcal{E}_{P_{\bar{n}_k}}(f)-\mathcal{E}_{\hat{P}_k}(f)\right|\leq\delta_k.$$

Also, by the inclusions of Proposition 6 and the definition of $\bar{n}_k=\bar{n}(\delta_k)$, we have on the event H that

$$\mathcal{F}_P(\delta_k)\subset\mathcal{F}_{P_{\bar{n}_k}}(2\delta_k).$$

Hence, for all $f\in\mathcal{F}_P(\delta_k)$,

$$\mathcal{E}_{P_{\bar{n}_k}}(f)\leq 2\delta_k \text{ and } \mathcal{E}_{\hat{P}_k}(f)\leq 3\delta_k.$$

This implies the inclusion

$$\mathcal{F}_P(\delta_k) \subset \hat{\mathcal{F}}_{k-1} \bigcap \mathcal{F}_{\hat{P}_k}(3\delta_k) = \hat{\mathcal{F}}_k.$$

On the other hand, since $\hat{\mathcal{F}}_k \subset \hat{\mathcal{F}}_{k-1}$, we have, for all $f \in \hat{\mathcal{F}}_k$,

$$\left| \mathcal{E}_{P_{\bar{n}_k}}(f) - \mathcal{E}_{\hat{P}_k}(f) \right| \leq \delta_k.$$

Since for all $f \in \hat{\mathcal{F}}_k$, $\mathcal{E}_{\hat{P}_k}(f) \leq 3\delta_k$, we also have $\mathcal{E}_{P_{\bar{n}_k}}(f) \leq 4\delta_k$. Thus, using again the inclusions of Proposition 6, we get

$$\hat{\mathcal{F}}_k \subset \mathcal{F}_{P_{\bar{n}_k}}(4\delta_k) \subset \mathcal{F}_P(8\delta_k),$$

proving the inclusions (2)

To prove the bound on $\nu(\delta)$, note that on the event H, where the inclusions (2) hold for all k such that $\delta_k \geq \delta$, we have

$$\hat{A}_k \subset A(8\delta_{k-1}).$$

Hence, on the event H,

$$\nu(\delta) \leq \sum_{\delta_k \geq \delta} \nu_k,$$

where

$$\nu_k := \sum_{j=1}^{\bar{n}_k} I_{A(8\delta_{k-1})}(X_j).$$

Clearly, ν_k is a binomial random variable with parameters $\bar{n}_k$ and $\pi(\delta_{k-1})$. Therefore, we have

$$\mathbb{P}\{\nu_k \geq s\} \leq \left(\frac{e\bar{n}_k \pi(\delta_{k-1})}{s} \right)^s$$

(see, e.g., Dudley, 1999, p. 16). Taking $s := et\bar{n}_k\pi(\delta_{k-1})$ yields

$$\mathbb{P}\Big\{ \nu_k \geq et\bar{n}_k\pi(\delta_{k-1}) \Big\} \leq \exp\{-\bar{n}_k\pi(\delta_{k-1})t\log t\}.$$

Applying the union bound, we get

$$\mathbb{P}\left\{ \nu(\delta) \geq et \sum_{\delta_k \geq \delta} \bar{n}_k\pi(\delta_{k-1}) \right\} \leq \mathbb{P}(H^c) + \sum_{\delta_k \geq \delta} \exp\{-\bar{n}_k\pi(\delta_{k-1})t\log t\}.$$

Since

$$\mathbb{P}(H^c) \leq \sum_{n \in M} \sum_{j \geq 0} e^{-t_j^{(n)}},$$

the result follows. ■

The simplest way to make the method data dependent is to replace in **Algorithm 1** the sample sizes $\bar{n}_k = \bar{n}(\delta_k)$ by their estimates $\hat{n}_k := \hat{n}(\delta_k)$, $k \geq 1$ and to redefine $\hat{P}_k$ in the iterative procedure for $\hat{A}_k, \hat{P}_k$ and $\hat{\mathcal{F}}_k$ as follows:

$$\hat{P}_k := \frac{1}{\hat{n}_k} \sum_{j=1}^{\hat{n}_k} I_{\hat{A}_k}(X_j)\delta_{X_j}.$$

This modification of **Algorithm 1** will be called **Algorithm 2**. The following statement can be proved quite similarly to Theorem 7 (using Proposition 6).

Recall the definition of the number of iterations L and also that $\nu(\delta)$ denotes the number of active examples used by the algorithm in the first L iterations.

Theorem 8 *With probability at least*

$$1-3\sum_{j\geq 0}\sum_{n\in M} e^{-t_j^{(n)}},$$

the following inclusions hold for the classes $\hat{\mathcal{F}}_k$ output by **Algorithm 2***: for all $k\geq 0$,*

$$\mathcal{F}_P(\delta_k)\subset \hat{\mathcal{F}}_k\subset \mathcal{F}_P(8\delta_k).$$

Moreover, for all $t\geq 1$ and for all $\delta\in(0,1]$, with probability at least

$$1-3\sum_{j\geq 0}\sum_{n\in M}\exp\{-t_j^{(n)}\}-\sum_{\delta_j\geq\delta}\exp\{-\tilde{n}(\delta_j)\pi(\delta_{j-1})t\log t\}$$

the following bound holds:

$$\nu(\delta)\leq et\sum_{\delta_j\geq\delta}\tilde{n}(\delta_j)\pi(\delta_{j-1}).$$

Note that in this version of the algorithm all the training examples X_j (not only the examples in the active sets $\hat{A}_k$) are used to determine the sample sizes $\hat{n}_k$. So, from this point of view, **Algorithm 2** can not be viewed as really "active". However, it is easy to see that in a more concrete framework of prediction problems (such as, for instance, the binary classification) one can modify the definitions of the localized Rademacher complexities and of the sample sizes $\hat{n}_k$ in such a way that they depend only on the design points, but not on the response variables (labels). Thus, in the cases when sampling the design points is "cheap" and only assigning the labels to them is "expensive" (which is a common motivational assumption in the literature on active learning), the algorithms of this type make some sense (see Section 4 for more details).

It is more interesting, however, that even in the abstract framework of empirical risk minimization it is possible to change the definition of Rademacher complexities and the estimates of the sample sizes based on them so that only the active examples that belong to the sets $\hat{A}_k$ are being used in the computation. We will describe such a data driven algorithm of active learning below.

Let $\delta_j:=2^{-j},\ j\geq 0$. As before, we will define iteratively data dependent function classes $\hat{\mathcal{F}}_k$ beginning with $\hat{\mathcal{F}}_0:=\mathcal{F}$ that provide estimates of the δ-minimal sets $\mathcal{F}_P(\delta)$ for sufficiently small values of δ and we set

$$\hat{A}_k:=\Big\{x:\ \sup_{f,g\in\hat{\mathcal{F}}_{k-1}}|f(x)-g(x)|>c\delta_k\Big\}\ \text{with some constant } c>0.$$

Define

$$\hat{P}_n^{(k)}:=n^{-1}\sum_{j=1}^n I_{\hat{A}_k}(X_j)\delta_{X_j}$$

and

$$\hat{R}_n^{(k)}(f):=n^{-1}\sum_{j=1}^n \varepsilon_j f(X_j)I_{\hat{A}_k}(X_j).$$

Denote

$$\hat{U}_n^{(k)} := \hat{K}\left[\sup_{f,g\in\hat{\mathcal{F}}_{k-1}}\left|\hat{R}_n^{(k)}(f-g)\right| + D_{\hat{P}_n^{(k)}}(\hat{\mathcal{F}}_{k-1})\sqrt{\frac{t_k^{(n)}}{n}} + \frac{t_k^{(n)}}{n}\right]$$

and define iteratively a nondecreasing data dependent sequence $\hat{n}_k$:

$$\hat{n}_k := \min\left\{n \in M, n \geq \hat{n}_{k-1} : \hat{U}_n^{(k)} \leq \frac{1}{2}\delta_{k+1}\right\}$$

with the initial condition $\hat{n}_0 := \inf M$.

Note that the following iterative relationships hold for the distribution dependent sample sizes $\bar{n}_k := \bar{n}(\delta_{k+1})$ and $\tilde{n}_k := \tilde{n}(\delta_{k+1})$:

$$\bar{n}_k = \min\left\{n \in M, n \geq \bar{n}_{k-1} : \bar{U}_n(\delta_k) \leq \frac{1}{2}\delta_{k+1}\right\},\ \ \bar{n}_0 := \inf M$$

and

$$\tilde{n}_k = \min\left\{n \in M, n \geq \tilde{n}_{k-1} : \tilde{U}_n(\delta_k) \leq \frac{1}{2}\delta_{k+1}\right\},\ \ \tilde{n}_0 := \inf M.$$

(which easily follows from the definitions of $\bar{n}(\delta), \tilde{n}(\delta)$).

We will write, for brevity, $\hat{P}_k := \hat{P}_{\hat{n}_k}^{(k)}$. With these definitions and notations, we can define $\hat{\mathcal{F}}_k$ iteratively exactly as before:

$$\hat{\mathcal{F}}_k := \hat{\mathcal{F}}_{k-1} \bigcap \mathcal{F}_{\hat{P}_k}(3\delta_k).$$

In short, the algorithm can be described as follows:

Algorithm 3

$\hat{\mathcal{F}}_0 := \mathcal{F}$;

for $k = 1, 2, \ldots,$

$\hat{A}_k := \left\{x : \sup_{f,g\in\hat{\mathcal{F}}_{k-1}} |f(x) - g(x)| > c\delta_k\right\}$;

$\hat{n}_k := \min\left\{n \in M, n \geq \hat{n}_{k-1} : \hat{U}_n^{(k)} \leq \frac{1}{2}\delta_{k+1}\right\}$;

$\hat{\mathcal{F}}_k := \hat{\mathcal{F}}_{k-1} \bigcap \mathcal{F}_{\hat{P}_k}(3\delta_k)$;

end

As before, we define

$$A(\delta) := \left\{x : \sup_{f,g\in\mathcal{F}(8\delta)} |f(x) - g(x)| > c\delta\right\}$$

and

$$\pi(\delta) := P(A(\delta)).$$

The properties of **Algorithm 3** are summarized in the following theorem.

Theorem 9 *There exist numerical constants c in the definition of the active sets $\hat{A}_k$, $\hat{K}$ in the definition of $\hat{U}_n^{(k)}$ and $\tilde{K}, \tilde{c}$ in the definition of the function $\tilde{U}_n$ such that the following holds. With probability at least*

$$1 - 3\sum_{j\geq 0}\sum_{n\in M} e^{-t_j^{(n)}},$$

the following inequalities and inclusions hold for all $k \geq 0$:

$$\bar{n}_k \leq \hat{n}_k \leq \tilde{n}_k,$$

$$\mathcal{F}_P(\delta_k) \subset \hat{\mathcal{F}}_k \subset \mathcal{F}_P(8\delta_k).$$

Moreover, for all $t \geq 1$, *with probability at least*

$$1 - 3\sum_{j\geq 0}\sum_{n\in M}\exp\{-t_j^{(n)}\} - \sum_{\delta_j \geq \delta}\exp\{-\tilde{n}(\delta_{j+1})\pi(\delta_{j-1})t\log t\}$$

the following bound holds:

$$\nu(\delta) \leq et\sum_{\delta_j\geq\delta}\tilde{n}(\delta_{j+1})\pi(\delta_{j-1}).$$

Proof There exists an event E of probability at least

$$1 - 3\sum_{n\in M}\sum_{j\geq 0} e^{-t_j^{(n)}}$$

on which the following holds. For all k and for all $n \in M$

$$\hat{K}\left[\sup_{f,g\in\mathcal{F}_P(8\delta_{k-1})}\left|R_n(f-g)\right| + D_{P_n}(\mathcal{F}_P(8\delta_{k-1}))\sqrt{\frac{t_k^{(n)}}{n}} + \frac{t_k^{(n)}}{n}\right] \leq \frac{1}{2}\tilde{U}_n(\delta_k) \tag{3}$$

and

$$\hat{K}\left[\sup_{f,g\in\mathcal{F}_P(\delta_{k-1})}\left|R_n(f-g)\right| + D_{P_n}(\mathcal{F}_P(\delta_{k-1}))\sqrt{\frac{t_k^{(n)}}{n}} + \frac{t_k^{(n)}}{n}\right] \geq 2\bar{U}_n(\delta_k) \tag{4}$$

with properly chosen constants in the definitions of the functions $\bar{U}_n, \tilde{U}_n$ and constant $\hat{K}$. At the same time, on the same event E, for all $n \in M, n \geq \bar{n}(\delta)$ and all $\sigma \geq \delta$,

$$\mathcal{F}_P(\sigma) \subset \mathcal{F}_{P_n}(2\sigma) \text{ and } \mathcal{F}_{P_n}(\sigma) \subset \mathcal{F}_P(2\sigma). \tag{5}$$

To construct such an event, first consider the event H of Proposition 6 on which the inclusions (5) hold. Then define

$$E_{k,n} := \left\{\hat{K}\left[\sup_{f,g\in\mathcal{F}_P(8\delta_{k-1})}\left|R_n(f-g)\right| + D_{P_n}(\mathcal{F}_P(8\delta_{k-1}))\sqrt{\frac{t_k^{(n)}}{n}} + \frac{t_k^{(n)}}{n}\right] \leq \frac{1}{2}\tilde{U}_n(\delta_k)\right\}$$

and

$$E'_{k,n} := \left\{\hat{K}\left[\sup_{f,g\in\mathcal{F}_P(\delta_{k-1})}\left|R_n(f-g)\right| + D_{P_n}(\mathcal{F}_P(\delta_{k-1}))\sqrt{\frac{t_k^{(n)}}{n}} + \frac{t_k^{(n)}}{n}\right] \geq 2\bar{U}_n(\delta_k)\right\}.$$

Using the "statistical version" of Talagrand's concentration inequality (Theorem 4) one can show that with a proper choice of $\hat{K}$ and the constants in the definitions of the functions $\bar{U}_n, \tilde{U}_n$,

$$\mathbb{P}(E_{n,k}) \leq 1 - e^{-t_k^{(n)}}, \ \ \mathbb{P}(E'_{n,k}) \geq 1 - e^{-t_k^{(n)}}$$

for all $k \geq 0$ and for all $n \in M$. Define

$$E := \bigcap_{k \geq 0, n \in M} (E_{n,k} \cap E'_{n,k}) \cap H.$$

Then

$$\mathbb{P}(E) \geq 1 - 3 \sum_{n \in M} \sum_{j \geq 0} e^{-t_j^{(n)}}$$

and all the desired properties hold on the event E.

We will now show by induction that, on the event E for $k = 0, 1, \ldots$

$$\bar{n}_k \leq \hat{n}_k \leq \tilde{n}_k,$$

$$\mathcal{F}_P(\delta_k) \subset \hat{\mathcal{F}}_k \subset \mathcal{F}_P(8\delta_k)$$

and also for $k = 1, 2, \ldots$ and for all $n \in M$

$$2\bar{U}_n(\delta_k) - \frac{\delta_{k+1}}{2} \leq \hat{U}_n^{(k)} \leq \frac{1}{2}\Big[\tilde{U}_n(\delta_k) + \frac{\delta_{k+1}}{2}\Big]. \tag{6}$$

By the definitions, the claims are obviously true for $k = 0$. Assume that they have been proved up to $k-1$. By this induction assumption, we have $\hat{\mathcal{F}}_{k-1} \subset \mathcal{F}_P(8\delta_{k-1})$ and, by the definition of the set $\hat{A}_k$,

$$\sup_{f,g \in \hat{\mathcal{F}}_{k-1}} \Big|\hat{R}_n^{(k)}(f-g)\Big| \leq \sup_{f,g \in \hat{\mathcal{F}}_{k-1}} \Big|R_n(f-g)\Big| + c\delta_k$$

and

$$D^2_{\hat{P}_n^{(k)}}(\hat{\mathcal{F}}_{k-1}) \leq D^2_{P_n}(\hat{\mathcal{F}}_{k-1}) + c^2\delta_k^2.$$

This implies the following upper bound on $\hat{U}_n^{(k)}$:

$$\hat{U}_n^{(k)} \leq \hat{K}\Bigg[\sup_{f,g \in \mathcal{F}_P(8\delta_{k-1})} \Big|R_n(f-g)\Big| + D_{P_n}(\mathcal{F}_P(8\delta_{k-1}))\sqrt{\frac{t_k^{(n)}}{n}} + \frac{t_k^{(n)}}{n} + c\delta_k + c\delta_k\sqrt{\frac{t_k^{(n)}}{n}}\Bigg].$$

Applying to the last term the inequality $ab \leq (a^2+b^2)/2$ and taking into account the fact that $\delta_k \leq 1$, it is easy to deduce from this that with c satisfying the condition

$$\hat{K}c + \hat{K}^2c^2/2 \leq 1/8,$$

we have

$$\hat{U}_n^{(k)} \leq \hat{K}\Bigg[\sup_{f,g \in \mathcal{F}_P(8\delta_{k-1})} \Big|R_n(f-g)\Big| + D_{P_n}(\mathcal{F}_P(8\delta_{k-1}))\sqrt{\frac{t_k^{(n)}}{n}} + \frac{t_k^{(n)}}{n}\Bigg] + \delta_{k+1}/4.$$

Quite similarly, using the inclusion $\mathcal{F}_P(\delta_{k-1}) \subset \hat{\mathcal{F}}_{k-1}$ that also holds under the induction assumption, one can show that with a proper choice of constant c in the definition of the set $\hat{A}_k$

$$\hat{U}_n^{(k)} \geq \hat{K}\Bigg[\sup_{f,g \in \mathcal{F}_P(\delta_{k-1})} \Big|R_n(f-g)\Big| + D_{P_n}(\mathcal{F}_P(\delta_{k-1}))\sqrt{\frac{t_k^{(n)}}{n}} + \frac{t_k^{(n)}}{n}\Bigg] - \delta_{k+1}/2.$$

Combining this with bounds (3) and (4) immediately implies (6).

Applying (6) to $n = \hat{n}_k$, we get

$$2\bar{U}_{\hat{n}_k}(\delta_k) - \frac{\delta_{k+1}}{2} \le \hat{U}^{(k)}_{\hat{n}_k} \le \frac{\delta_{k+1}}{2},$$

which yields

$$\bar{U}_{\hat{n}_k}(\delta_k) \le \frac{\delta_{k+1}}{2}.$$

We also have $\hat{n}_k \ge \hat{n}_{k-1} \ge \bar{n}_{k-1}$ (by the induction assumption). By the definition of $\bar{n}_k$, this implies that $\hat{n}_k \ge \bar{n}_k$.

On the other hand, denote $\hat{n}_k^-$ the element of the ordered set M preceding $\hat{n}_k$. We will use inequality (6) with $n = \hat{n}_k^-$. It gives

$$\hat{U}^{(k)}_{\hat{n}_k^-} \le \frac{1}{2}\Big[\tilde{U}_{\hat{n}_k^-}(\delta_k) + \frac{\delta_{k+1}}{2}\Big]. \tag{7}$$

If it happened that $\hat{n}_k^- < \hat{n}_{k-1}$, then we must have $\hat{n}_k = \hat{n}_{k-1}$, which, by the induction assumption, implies that $\hat{n}_k = \hat{n}_{k-1} \le \tilde{n}_{k-1} \le \tilde{n}_k$. If $\hat{n}_k^- \ge \hat{n}_{k-1}$, then the definition of $\hat{n}_k$ implies that

$$\hat{U}^{(k)}_{\hat{n}_k^-} > \frac{\delta_{k+1}}{2},$$

which together with (7) implies that

$$\tilde{U}_{\hat{n}_k^-}(\delta_k) > \frac{\delta_{k+1}}{2}.$$

But, if $\hat{n}_k > \tilde{n}_k$, then $\hat{n}_k^- \ge \tilde{n}_k$, which would imply that

$$\tilde{U}_{\tilde{n}_k}(\delta_k) > \frac{\delta_{k+1}}{2}$$

(since for all δ, $\tilde{U}_n(\delta)$ is a nonincreasing function of n). The last inequality contradicts the definition of $\tilde{n}_k$ implying that $\hat{n}_k \le \tilde{n}_k$.

The proof of the inclusions

$$\mathcal{F}_P(\delta_k) \subset \hat{\mathcal{F}}_k \subset \mathcal{F}_P(8\delta_k)$$

and the derivation of the bound on $\nu(\delta)$ repeat the argument of Theorem 7. ■

Note that in the bounds on $\nu(\delta)$ of theorems 7, 8 and 9 one can replace functions $\pi(\delta)$, $\bar{n}(\delta)$ and $\tilde{n}(\delta)$ by arbitrary upper bounds (with the same change in the bounds on the probability).

As soon as $\pi(\delta) \to 0$ as $\delta \to 0$, the upper bounds on $\nu(\delta)$ show that, in the case of active learning, there is a reduction of the number of training examples needed to achieve a desired accuracy of learning comparing with passive learning. In the next section, we explore in some detail what is happening in the case of binary classification.

4. Active Learning in Binary Classification

Let (X,Y) be a random couple with values in $S\times\{-1,1\}$ and with distribution P, where $(S,\mathcal{A})$ is an arbitrary measurable space. In binary classification problems, the first component X is viewed as an observable instance and the second component Y is an unobservable "label". The value of Y is to be predicted based on an observation of X and on the training data $(X_1,Y_1),\dots,(X_n,Y_n)$ consisting of n independent copies of (X,Y). Measurable functions $g:S\mapsto\{-1,1\}$ are called (binary) classifiers. Let $\ell:\{-1,1\}\times\{-1,1\}\mapsto\{0,1\}$ be the binary loss function $\ell(y,u):=I(y\neq u)$, and, as before, $(\ell\bullet g)(x,y):=\ell(y,g(x))$ be the "loss" of classifier g for the example $(x,y)\in S\times\{-1,1\}$. The quantity

$$P(\ell\bullet g)=P\{(x,y):y\neq g(x)\}=\mathbb{P}\{Y\neq g(X)\}$$

is called the generalization error, or the risk of g. We still denote $\eta(x):=\mathbb{E}(Y|X=x)$ the regression function. It is well known that the minimum of the generalization error over the set of all binary classifiers is attained at the Bayes classifier

$$g_*(x)=\mathrm{sign}(\eta(x)).$$

We will assume in what follows that $\mathcal{G}$ is a class of binary classifiers such that $g_*\in\mathcal{G}$.

For a binary classifier g, define its excess risk as

$$\mathcal{E}_P(\ell\bullet g):=P(\ell\bullet g)-P(\ell\bullet g_*).$$

The following formula is well known (see, e.g., Devroye et al., 1996, Theorem 2.2).

$$\mathcal{E}_P(\ell\bullet g)=\int_{\{g\neq g_*\}}|\eta(x)|\Pi(dx), \tag{8}$$

where Π is the marginal distribution of X.

A standard approach to learning the Bayes classifier is based on the empirical risk minimization:

$$\hat{g}:=\mathrm{argmin}_{g\in\mathcal{G}}P_n(\ell\bullet g)=\mathrm{argmin}_{g\in\mathcal{G}}P_n\{(x,y):y\neq g(x)\}=$$

$$\mathrm{argmin}_{g\in\mathcal{G}}n^{-1}\sum_{j=1}^{n}I(Y_j\neq g(X_j)),$$

where P_n denotes the empirical distribution based on the training data $(X_1,Y_1),\dots,(X_n,Y_n)$ (we will also use the notation Π_n for the empirical distribution based on $(X_1,\dots,X_n)$).

If $\mathcal{F}:=\ell\bullet\mathcal{G}:=\{\ell\bullet g:g\in\mathcal{G}\}$ denotes the loss class, then we are in the framework of abstract empirical risk minimization of sections 2,3 and general results of these sections can be now specialized for the classification context.

It is natural to characterize the quality of the classifier $\hat{g}$ in terms of its excess risk $\mathcal{E}_P(\ell\bullet\hat{g})$ and to study how it depends on the complexity of the class $\mathcal{G}$ as well as on the complexity of the classification problem itself. The simplest complexity assumption on the class $\mathcal{G}$ is that it is a VC-class of binary functions of VC-dimension V (in other words, $\mathcal{C}:=\Big\{\{x:g(x)=+1:g\in\mathcal{G}\}\Big\}$ is a VC-class of sets of VC-dimension V). Under this assumption, a well known result, essentially due to Vapnik and Chervonenkis, is that, for some constant $K>0$ and for all $t>0$, with probability at least $1-e^{-t}$

$$\mathcal{E}_P(\ell\bullet\hat{g})\leq K\left[\sqrt{\frac{V}{n}}+\sqrt{\frac{t}{n}}\right].$$

In principle, this bound is minimax optimal, but it can be significantly improved for special families of distributions P under further assumptions on the complexity of the classification problem. For instance, the following **Massart's low noise assumption** is frequently used: for some constant $h \in (0,1]$

$$|\eta(x)| \geq h,\ x \in S.$$

The parameter h is a characteristic of the level of noise in binary labels Y_j. In other words, it is a simple measure of complexity of a binary classification problem. The following theorem is a version of the result proved by Massart and Nedelec (2006):

Theorem 10 *There exists a constant $K > 0$ such that, for all $t > 0$, with probability at least $1 - e^{-t}$*

$$\mathcal{E}_P(\ell \bullet \hat{g}) \leq K\left[\frac{V}{nh}\log\Big(\frac{nh^2}{V}\Big) + \frac{t}{nh}\right] \bigwedge \left[\sqrt{\frac{V}{n}} + \sqrt{\frac{t}{n}}\right].$$

This upper bound on the excess risk is optimal in a minimax sense (as it was also shown by Massart and Nedelec, 2006). However, it still can be refined using the following function τ (which is a local version of Alexander's **capacity function** introduced in the 80s and used in the theory of ratio type empirical processes, see Giné and Koltchinskii, 2006, and references therein). Define

$$\mathcal{G}_\delta := \{g \in \mathcal{G} : \Pi\{x : g(x) \neq g_*(x)\} \leq \delta\}$$

and let

$$\tau(\delta) := \frac{\Pi\Big\{x | \exists g \in \mathcal{G}_\delta : g(x) \neq g_*(x)\Big\}}{\delta}.$$

Clearly, the set $\mathcal{G}_\delta$ consists of the classifiers from $\mathcal{G}$ that are in a neighborhood of size δ of the Bayes classifier g_* and the set

$$D_\delta := \Big\{x | \exists g \in \mathcal{G}_\delta : g(x) \neq g_*(x)\Big\}$$

consists of all the points x such that there exists a classifier g in the neighborhood $\mathcal{G}_\delta$ that "disagrees" with the Bayes classifier at x. The function $\tau(\delta)$ is always upper bounded by $\frac{1}{\delta}$. However, if it happens that the measure Π of the "disagreement set" D_δ is small when δ is small, then $\tau(\delta)$ might grow slower than $\frac{1}{\delta}$ as $\delta \to 0$, or even it can be bounded by a constant. If $\mathcal{C} := \{\{g = +1\} : g \in \mathcal{G}\}$ and $C_* := \{g_* = +1\}$, then

$$\tau(\delta) = \frac{\Pi\Big(\bigcup_{C \in \mathcal{C}, \Pi(C \triangle C_*) \leq \delta}(C \triangle C_*)\Big)}{\delta},$$

so, roughly, $\tau(\delta)$ shows how many disjoint sets $C \triangle C_*$ of "size" δ can be "packed" in the union of all such sets. For instance, if $\mathcal{C}$ is a class of convex sets in $[0,1]^d$, Π is the Lebesgue measure in $[0,1]^d$ and $C_* \in \mathcal{C}, \Pi(C_*) > 0$, then it can be shown that τ is uniformly bounded by a constant, see Giné and Koltchinskii (2006). A more detailed analysis of disagreement sets, capacity functions and their connections to the geometry of the class $\mathcal{G}$ can be found in the paper by Friedman (2009).

The following result was proved by Giné and Koltchinskii (2006).

Theorem 11 *There exists a constant $K > 0$ such that, for all $t > 0$, with probability at least $1 - e^{-t}$*

$$\mathcal{E}_P(\ell \bullet \hat{g}) \le K\left[\frac{V}{nh}\log\tau\Big(\frac{V}{nh^2}\Big) + \frac{t}{nh}\right] \bigwedge \left[\sqrt{\frac{V}{n}} + \sqrt{\frac{t}{n}}\right].$$

Clearly, this result implies the theorem of Massart and Nedelec (since $\tau(\delta) \le \frac{1}{\delta}$). The proof is based on applying subtle bounds for empirical processes (see Giné and Koltchinskii, 2006) to compute the excess risk bound $\bar{\delta}_n$ of Section 2. Then, the general result of Theorem 1 (see also Proposition 2) can be used to bound the excess risk.

The case when $\tau(\delta)$ is uniformly bounded from above by a constant τ_0 is of special interest. In this case, with probability at least $1 - e^{-t}$,

$$\mathcal{E}_P(\ell \bullet \hat{g}) \le K\left[\frac{V}{nh}\log\tau_0 + \frac{t}{nh}\right],$$

so the main term of the bound is of the order $O(\frac{V}{nh})$ and it does not contain logarithmic factors depending on n and h. It will be convenient for our purposes to phrase this result in a slightly different form. Namely, given $\delta \in (0,1)$ and $\alpha \in (0,1)$, denote

$$n(\delta,\alpha) := \inf\left\{n : \mathbb{P}\{\mathcal{E}_P(\hat{g}_n) \ge \delta\} \le \alpha\right\}.$$

Then

$$n(\delta,\alpha) \le K\left(\left[\frac{V}{\delta h}\log\tau_0 + \frac{\log(1/\alpha)}{\delta h}\right] \bigwedge \left[\frac{V}{\delta^2} + \frac{\log(1/\alpha)}{\delta^2}\right]\right).$$

The quantity $n(\delta,\alpha)$ shows how many training examples are needed to make the excess risk of the classifier $\hat{g}$ smaller than δ with a guaranteed probability of at least $1-\alpha$. It characterizes the sample complexity of passive learning. In the case of empirical risk minimization over a VC-class with a bounded capacity function τ, the sample complexity is of the order $O(\frac{V}{h}\frac{1}{\delta})$.

The role of the capacity function is rather modest in the case of passive learning since it only allows one to refine the excess risk and the sample complexity bounds by making the logarithmic factors more precise. However, the capacity function τ happened to be of crucial importance in the analysis of active learning methods of binary classification. This function was rediscovered in active learning literature and its supremum is being used there under the name of **disagreement coefficient**, see, for example, Hanneke (2009a,b) and references therein.

We will describe an active learning algorithm that is a specialized version of more abstract **Algorithm 3** of Section 3. As before, we denote $\delta_j := 2^{-j},\ j \ge 0$ and choose a set $M \subset \mathbb{N}$ of natural numbers as well as nonnegative real numbers $t_k^{(n)},\ n \in M, k \ge 0$.

Given a class $\mathcal{G}$ of binary classifiers, denote

$$\mathcal{G}_P(\delta) := \left\{g : \mathcal{E}_P(\ell \bullet g) \le \delta\right\},\ \delta > 0.$$

These sets will be called δ-minimal sets of the true risk. Clearly, if $\mathcal{F} = \ell \bullet \mathcal{G}$, then under the notations of Section 2

$$\mathcal{F}_P(\delta) = \left\{\ell \bullet g : g \in \mathcal{G}_P(\delta)\right\}.$$

In principle, one can directly use **Algorithm 3** for the class $\mathcal{F}$. However, we will modify it slightly in order to adapt it to the binary classification framework.

We will define iteratively data dependent function classes $\hat{\mathcal{G}}_k$ that provide estimates of the δ-minimal sets $\mathcal{G}_P(\delta)$. and also a nondecreasing data dependent sequence of estimated sample sizes $\hat{n}_k$. It will be assumed that we have an access to an algorithm that, given a discrete measure Q on $S\times\{-1,1\}$ and $\delta>0$, computes the δ-minimal set $\mathcal{G}_Q(\delta)$ of Q.

Several definitions and notations will be needed. Note that for the binary loss ℓ, for all binary classifiers g_1,g_2 and for all $\delta\in(0,1)$, the condition $|\ell(y,g_1(x))-\ell(y,g_2(x))|\geq\delta$ is equivalent to the condition $g_1(x)\neq g_2(x)$. This leads to the following definition of sets $\hat{A}_k$ (that are subsets of S, not of $S\times\{-1,1\}$). Assuming that $\hat{\mathcal{G}}_{k-1}$ has been already defined, let

$$\hat{A}_k:=\Big\{x:\exists g_1,g_2\in\hat{\mathcal{G}}_{k-1},g_1(x)\neq g_2(x)\Big\}$$

be the set of all the points where at least two classifiers in $\hat{\mathcal{G}}_{k-1}$ disagree with each other. This set will be used as a set of active design points at the k-th iteration.

Next define active empirical distributions based on the unlabeled examples $\{X_j\}$ and on the labeled examples $\{(X_j,Y_j)\}$:

$$\hat{\Pi}_n^{(k)}:=n^{-1}\sum_{j=1}^n I_{\hat{A}_k}(X_j)\delta_{X_j}$$

and

$$\hat{P}_n^{(k)}:=n^{-1}\sum_{j=1}^n I_{\hat{A}_k}(X_j)\delta_{(X_j,Y_j)}$$

For simplicity, we will also use the notation $\hat{P}_k:=\hat{P}_{\hat{n}_k}^{(k)}$. Let

$$\hat{D}_n^{(k)}:=\frac{1}{2}\sup_{g_1,g_2\in\hat{\mathcal{G}}_{k-1}}\Big(\hat{\Pi}_n^{(k)}(g_1-g_2)^2\Big)^{1/2}$$

be the $L_2(\hat{\Pi}_n^{(k)})$-diameter of the class $\hat{\mathcal{G}}_{k-1}$. Note that, if we literally followed the definitions of Section 3, we would have to define the diameter as

$$\sup_{g_1,g_2\in\hat{\mathcal{G}}_{k-1}}\Big(\hat{P}_n^{(k)}(\ell\bullet g_1-\ell\bullet g_2)^2\Big)^{1/2}.$$

However, it is easy to check that for all $(x,y)\in S\times\{-1,1\}$ and all binary classifiers g_1,g_2

$$(\ell\bullet g_1)(x,y)-(\ell\bullet g_2)(x,y)=\frac{1}{2}y(g_2(x)-g_1(x)),$$

which justifies the new definition. This simple identity also implies that the function $\phi_n(\delta)$, defined in Section 2 and used in the construction of the excess risk bounds, can be upper bounded as follows:

$$\phi_n(\delta)\leq 2\mathbb{E}\sup_{f_1,f_2\in\mathcal{F}(\delta)}|R_n(f_1-f_2)|=2\mathbb{E}\sup_{g_1,g_2\in\mathcal{G}(\delta)}|R_n(\ell\bullet g_1-\ell\bullet g_2)|=$$

$$\mathbb{E}\sup_{g_1,g_2\in\mathcal{G}(\delta)}\Big|n^{-1}\sum_{j=1}^n\varepsilon_jY_j(g_2(X_j)-g_1(X_j))\Big|,$$

where at the beginning we used the symmetrization inequality (see, e.g., van der Vaart and Wellner, 1996). Note that, conditionally on $(X_1,Y_1),\dots,(X_n,Y_n)$, the distribution of the random vector $(\varepsilon_1 Y_1,\dots,\varepsilon_n Y_n)$ is the same as the distribution of $(\varepsilon_1,\dots,\varepsilon_n)$. Because of this,

$$\phi_n(\delta) \le \mathbb{E} \sup_{g_1,g_2\in\mathcal{G}(\delta)} \left| n^{-1} \sum_{j=1}^n \varepsilon_j Y_j (g_2(X_j)-g_1(X_j)) \right| =$$

$$\mathbb{E}\mathbb{E}\left(\sup_{g_1,g_2\in\mathcal{G}(\delta)} \left| n^{-1} \sum_{j=1}^n \varepsilon_j Y_j (g_2(X_j)-g_1(X_j)) \right| \Big| (X_1,Y_1),\dots,(X_n,Y_n) \right) =$$

$$\mathbb{E}\mathbb{E}\left(\sup_{g_1,g_2\in\mathcal{G}(\delta)} \left| n^{-1} \sum_{j=1}^n \varepsilon_j (g_2(X_j)-g_1(X_j)) \right| \Big| (X_1,Y_1),\dots,(X_n,Y_n) \right) =$$

$$\mathbb{E} \sup_{g_1,g_2\in\mathcal{G}(\delta)} |R_n(g_1-g_2)|.$$

This simple observation allows one to replace the Rademacher complexities for the loss class $\mathcal{F} = \ell \bullet \mathcal{G}$ by the Rademacher complexities for the class $\mathcal{G}$ itself (and the proofs of the excess risk bounds and other results cited in Section 2 go through with no changes). Of course, the same applies to all the constructions and the results of Section 3.

Because of this, we now define the Rademacher complexity based only on the "active" examples as

$$\hat{R}_n^{(k)} := \sup_{g_1,g_2\in\hat{\mathcal{G}}_{k-1}} \left| n^{-1} \sum_{j=1}^n \varepsilon_j (g_1-g_2)(X_j) I_{\hat{A}_k}(X_j) \right|.$$

Finally, denote

$$\hat{U}_n^{(k)} := \hat{K}\left[\hat{R}_n^{(k)} + \hat{D}_n^{(k)} \sqrt{\frac{t_k^{(n)}}{n}} + \frac{t_k^{(n)}}{n} \right].$$

With these definitions and notations, we can now introduce the following modification of **Algorithm 3** of Section 3.

Algorithm 4
$\hat{\mathcal{G}}_0 := \mathcal{G}$;
for $k = 1,2,\dots,$
$\hat{A}_k := \left\{ x : \exists g_1,g_2 \in \hat{\mathcal{G}}_{k-1}, g_1(x) \ne g_2(x) \right\}$;
$\hat{n}_k := \min\left\{ n \in M, n \ge \hat{n}_{k-1} : \hat{U}_n^{(k)} \le \frac{1}{2}\delta_{k+1} \right\}$;
$\hat{\mathcal{G}}_k := \hat{\mathcal{G}}_{k-1} \bigcap \mathcal{G}_{\hat{P}_k}(3\delta_k)$;
end

Remark 12 *One can also use in* **Algorithm 4** *the Rademacher complexities defined as follows:*

$$\hat{R}_n^{(k)} := \sup_{g_1,g_2\in\hat{\mathcal{G}}_{k-1}} \left| n^{-1} \sum_{j=1}^n \varepsilon_j (g_1-g_2)(X_j) \right|.$$

In this case, not only the active design points, but all the design points X_j are used to compute the Rademacher complexities and to estimate the sample sizes $\hat{n}_k$. Note, however, that the labels Y_j are not involved in this computation, so, the algorithm still can be viewed as "active". The resulting algorithm is a modification of **Algorithm 3** *from Section 3.*

In the case when $\mathcal{G}$ is a VC-class of VC-dimension V, we will choose $M := \{2^k : k \geq 0\}$. We will also define

$$t_k^{(n)} := \log(1/\alpha) + 2\log(k+1) + 2\log(\log_2 n + 1) + \log(24). \tag{9}$$

This leads to the following result that is a corollary of Theorem 9.

Corollary 13 *Let $\delta \in (0,1)$. Suppose that Massart's low noise assumption holds with some $h \in (0,1)$. Suppose that*

$$\tau_0 := \sup_{u \in (0,1]} \tau(u) < +\infty.$$

Then there exists an event of probability at least $1-\alpha$ such that the following inclusions hold for the classes $\hat{\mathcal{G}}_k$ output by **Algorithm 4***: for all $k \geq 0$,*

$$\mathcal{G}_P(\delta_k) \subset \hat{\mathcal{G}}_k \subset \mathcal{G}_P(8\delta_k). \tag{10}$$

Also with probability at least $1-\alpha$, the following bound on the number $\nu(\delta)$ of active training examples used by **Algorithm 4** *in the first $L = \left\lfloor \log_2(1/\delta) \right\rfloor$ iterations holds with some numerical constant $C > 0$:*

$$\nu(\delta) \leq C \frac{\tau_0 \log(1/\delta)}{h^2} \left[V \log \tau_0 + \log(1/\alpha) + \log\log(1/\delta) + \log\log(1/h) \right].$$

In particular, it means that with probability at least $1-\alpha$

$$\mathcal{G}_P(\delta/2) \subset \hat{\mathcal{G}}_L \subset \mathcal{G}_P(16\delta).$$

Proof We only sketch the proof here, the missing details are not very complicated. The result follows from Theorem 9, more precisely, from its modified version that takes into account the slight changes we made in the definition of the Rademacher complexities. The inclusions (10) follow from this theorem in a straightforward way. To prove the bound on $\nu(\delta)$, one has first to bound the quantity $\bar{\delta}_n$. This computation was essentially done by Giné and Koltchinskii (2006) (it actually leads to the bound of Theorem 11). Namely, with some constant C_1,

$$\bar{\delta}_n \leq C_1 \left[\frac{V}{nh} \log \tau\Big(\frac{V}{nh^2}\Big) + \frac{\log(1/\alpha) + \log\log n}{nh} \right] \bigwedge \left[\sqrt{\frac{V}{n}} + \sqrt{\frac{\log(1/\alpha) + \log\log n}{n}} \right].$$

As a result, the following upper bound on $\tilde{n}(\sigma), \sigma \geq \delta$ holds with some constant K_1 :

$$\tilde{n}(\sigma) \leq K_1 \Bigg(\left[\frac{V}{\sigma h} \log \tau_0 + \frac{\log(1/\alpha) + \log\log_2(1/\delta) + \log\log(1/h)}{\sigma h} \right] \bigwedge$$

$$\left[\frac{V}{\sigma^2} + \frac{\log(1/\alpha) + \log\log_2(1/\delta)}{\sigma^2} \right] \Bigg).$$

Note that, under Massart's low noise assumption, formula (8) for the excess risk implies that for all binary classifiers g

$$\mathcal{E}(\ell \bullet g) \geq h \Pi\{x : g(x) \neq g_*(x)\}.$$

Hence

$$\mathcal{F}(\sigma) \subset \Big\{\ell \bullet g : g \in \mathcal{G}_{\sigma/h}\Big\}.$$

For the sets $A(\sigma)$ used in Theorems 9, this implies the following:

$$A(\sigma) = \Big\{(x,y) : \sup_{f_1,f_2\in\mathcal{F}(8\sigma)} |f_1(x,y) - f_2(x,y)| > c\sigma\Big\} \subset$$

$$\Big\{(x,y) : \sup_{g_1,g_2\in\mathcal{G}(8\sigma/h)} |(\ell \bullet g_1)(x,y) - (\ell \bullet g_2)(x,y)| > c\sigma\Big\} =$$

$$\Big\{x : \exists g_1, g_2 \in \mathcal{G}(8\sigma/h) : g_1(x) \neq g_2(x)\Big\} \times \{-1,1\} = \Big\{x : \exists g \in \mathcal{G}(8\sigma/h) : g(x) \neq g_*(x)\Big\} \times \{-1,1\}$$

(we used the assumption that $g_* \in \mathcal{G}$ and, hence, $g_* \in \mathcal{G}(8\sigma/h)$). This implies, by the definitions of the functions π and τ, that

$$\pi(\sigma) = P(A(\sigma)) \leq \Pi\Big(\Big\{x : \exists g \in \mathcal{G}(8\sigma/h) : g(x) \neq g_*(x)\Big\}\Big) \leq \frac{8\sigma}{h}\tau(8\sigma/h).$$

Using the definition of τ_0, we conclude that for all $\sigma \geq \delta$ $\pi(\sigma) \leq \frac{8\tau_0}{h}\sigma$. It remains to substitute the bounds on $\tilde{n}(\sigma)$ and $\pi(\sigma)$ into the bound on $\nu(\delta)$ of Theorem 9

$$\nu(\delta) \leq et \sum_{\delta_j \geq \delta} \tilde{n}(\delta_{j+1})\pi(\delta_{j-1}),$$

say, with $t = e$. This gives

$$\nu(\delta) \leq e^2 \sum_{\delta_j \geq \delta} K_1\left(\left[\frac{V}{\delta_{j+1}h}\log\tau_0 + \frac{\log(1/\alpha) + \log\log_2(1/\delta) + \log\log(1/h)}{\delta_{j+1}h}\right]\frac{8\tau_0}{h}\delta_{j-1},\right.$$

which is bounded from above by

$$C\frac{\tau_0\log(1/\delta)}{h^2}\Big[V\log\tau_0 + \log(1/\alpha) + \log\log(1/\delta) + \log\log(1/h)\Big].$$

with a properly chosen numerical constant C. Also, it easily follows from the probability estimates of Theorem 9 that the above bound on $\nu(\delta)$ holds with probability at least $1-\alpha$. ■

Finally, we discuss the properties of **Algorithm 4** under **Tsybakov's low noise assumption**. Namely, we assume that for some $\gamma > 0$, for some constant B and for all $t > 0$

$$\Pi\{x : |\eta(x)| \leq t\} \leq Bt^{\gamma}.$$

It is well known that under this assumption the following bound on the excess risk holds for an arbitrary classifier g :

$$\mathcal{E}_P(\ell \bullet g) \geq c\Pi^{\kappa}(\{g \neq g_*\}),$$

where $\kappa = \frac{1+\gamma}{\gamma}$ and c is a constant that depends on B, κ. We will assume in this case that $\mathcal{G}$ is not necessarily a VC-class, but it can be more massive. For instance, denote $N(\mathcal{G}; L_2(\Pi_n); \varepsilon)$ the

minimal number of $L_2(\Pi_n)$-balls of radius ε needed to cover $\mathcal{G}$ and suppose that these covering numbers satisfy the condition:

$$\log N(\mathcal{G}; L_2(\Pi_n); \varepsilon) \le \left(\frac{A}{\varepsilon}\right)^{2\rho}, \ \varepsilon > 0.$$

for some $\rho \in (0,1]$ and some constant $A > 0$. Then, the following upper bound on the excess risk of an empirical risk minimizer $\hat{g}$ holds with probability at least $1 - e^{-t}$:

$$\mathcal{E}_P(\ell \bullet \hat{g}) \le K\left(\left(\frac{1}{n}\right)^{-\kappa/(2\kappa+\rho-1)} + \left(\frac{t}{n}\right)^{\kappa/(2\kappa-1)}\right),$$

where K is a constant depending on κ, ρ, A, B. The bounds of this type were first proved by Tsybakov (2004) (see also Koltchinskii, 2006, 2008). It easily follows from this bound that in order to achieve the excess risk of order δ one needs $O\left(\delta^{-2+(1-\rho)/\kappa}\right)$ training examples.

We will now consider **Algorithm 4** with $M := \{2^k : k \ge 0\}$, and with the real numbers $t_k^{(n)}$ defined by (9).

This leads to the following result that is also a corollary of Theorem 9.

Corollary 14 *Let $\delta \in (0,1)$. Suppose that Tsybakov's low noise assumption holds with some $\gamma > 0$ and $B > 0$. Let $\kappa := \frac{1+\gamma}{\gamma}$. Suppose that*

$$\tau_0 := \sup_{u \in (0,1]} \tau(u) < +\infty.$$

Then there exists an event of probability at least $1 - \alpha$ such that the following inclusions hold for the classes $\hat{\mathcal{G}}_k$ output by **Algorithm 4***: for all k with $\delta_k \ge \delta$,*

$$\mathcal{G}_P(\delta_k) \subset \hat{\mathcal{G}}_k \subset \mathcal{G}_P(8\delta_k).$$

Also with probability at least $1 - \alpha$, the following bound on the number $\nu(\delta)$ of active training examples used by **Algorithm 4** *holds with some constant $C > 0$ depending on κ, ρ, A, B :*

$$\nu(\delta) \le C\tau_0\left[\delta^{-2+(2-\rho)/\kappa} + \delta^{-2+2/\kappa}(\log(1/\alpha) + \log\log(1/\delta))\right].$$

The proof is similar to that of Corollary 13. In this case, the improvement comparing with passive learning is by a factor $\delta^{1/\kappa}$.

Remark 15 *Alternatively, one can assume that the active learning algorithm stops as soon as the specified number of active examples, say, n has been achieved. If $\hat{L}$ denotes the number of iterations needed to achieve this target, then $8\delta_{\hat{L}}$ is an upper bound on the excess risk of the classifiers from the set $\hat{\mathcal{G}}_{\hat{L}}$. Under the assumptions of Corollary 13, inverting the bound on $\nu(\delta)$ easily gives that $\delta_{\hat{L}}$ is upper bounded by*

$$\exp\left\{-\beta\frac{nh^2}{C_2\tau_0}\right\},$$

where

$$\beta := \frac{1}{V\log\tau_0 \vee \log(1/\alpha) \vee \log(nh^2/C_2\tau_0) \vee \log\log(1/h)}$$

with some numerical constant C_2. Thus, the excess risk of such classifiers tends to zero exponentially fast as $n \to \infty$. This is the form in which the excess risk bounds in active learning are usually stated in the literature (see, e.g., Hanneke, 2009a,b). In fact, this is a refinement of the bounds of Hanneke that were proved for somewhat different active learning algorithms (see Hanneke, 2009b, Theorems 4, 5). Similarly, under the conditions of Corollary 14, the bound on $\delta_{\hat{L}}$ becomes

$$\left(\frac{\tau_0}{n}\right)^{\kappa/(2\kappa+\rho-2)} \bigvee \left(\frac{\tau_0(\log(1/\alpha)+\log\log n)}{n}\right)^{\kappa/(2\kappa-2)}.$$

(compare with Theorem 6 by Hanneke, 2009b).

Remark 16 *Although we concentrated in this section only on binary classification problems, the active learning algorithms described in Section 3 can be also used in the context of multiclass classification and some other problems (e.g., estimation of non-smooth regression function and estimation of level sets of a probability density). Recall that in the framework of prediction with a general loss function ℓ described in the Introduction,*

$$\mathcal{F} := \ell \bullet \mathcal{G} := \Big\{\ell \bullet g : g \in \mathcal{G}\Big\}.$$

Following an idea of Beygelzimer et al. (2009), one can now replace the disagreement set $\hat{A}_k$ for the class $\hat{\mathcal{F}}_{k-1} = \ell \bullet \hat{\mathcal{G}}_{k-1}$ involved in **Algorithm 3** *by a larger set*

$$\hat{A}_k^+ := \left\{(x,y) : \exists g_1, g_2 \in \hat{\mathcal{G}}_{k-1} \ \sup_{y' \in T} |\ell(y', g_1(x)) - \ell(y', g_2(x))| > c\delta_k\right\} =$$

$$\left\{x : \exists g_1, g_2 \in \hat{\mathcal{G}}_{k-1} \ \sup_{y \in T} |\ell(y, g_1(x)) - \ell(y, g_2(x))| > c\delta_k\right\} \times T.$$

This leads to the following modification of **Algorithm 3***:*

Algorithm 5
$\hat{\mathcal{G}}_0 := \mathcal{G}$;
for $k = 1, 2, \ldots,$
$\hat{A}_k^+ := \left\{x : \exists g_1, g_2 \in \hat{\mathcal{G}}_{k-1} \ \sup_{y \in T} |\ell(y, g_1(x)) - \ell(y, g_2(x))| > c\delta_k\right\} \times T.$
$\hat{n}_k := \min\Big\{n \in M, n \geq \hat{n}_{k-1} : \hat{U}_n^{(k)} \leq \frac{1}{2}\delta_{k+1}\Big\}$;
$\hat{\mathcal{G}}_k := \hat{\mathcal{G}}_{k-1} \bigcap \mathcal{G}_{\hat{P}_k}(3\delta_k)$;
end

The Rademacher complexities of classes $\hat{\mathcal{F}}_k = \ell \bullet \hat{\mathcal{G}}_k$ (the quantities $\hat{U}_n^{(k)}$) as well as active empirical measures $\hat{P}_k$ involved in this algorithm are now based on active sets $\hat{A}_k^+$. Clearly, only the labels of active examples are used in this version of the algorithm. If now we define

$$\pi(\delta) := \Pi\Big(\Big\{x : \exists g_1, g_2 \in \mathcal{G}_P(8\delta) \ \sup_{y \in T} |\ell(y, g_1(x)) - \ell(y, g_2(x))| > c\delta\Big\}\Big),$$

it is very easy to check that the statement of Theorem 9 still holds for such a modification of the algorithm. At the same time, it is not clear at this point whether a modified definition of disagreement coefficient in the paper by Beygelzimer et al. (2009) can be used to analyze the properties of

active learning algorithms of this type and whether it is possible to extend such an analysis beyond classification and similar problems.

Acknowledgments

The author is thankful to anonymous referees for a number of helpful comments and corrections.

References

M.–F. Balcan, S. Hanneke and J. Wortman. The true sample complexity of active learning. In: *Proc. 21st Annual Conference on Learning Theory (COLT 2008),* pages 45–56, 2008.

M.–F. Balcan, A. Beygelzimer and J. Langford. Agnostic active learning. *J. of Computer and System Sciences,* 75(1):78–89, 2009.

P. Bartlett, S. Boucheron and G. Lugosi. Model selection and error estimation. *Machine Learning,* 48:85–113, 2002.

P. Bartlett, O. Bousquet and S. Mendelson. Local Rademacher complexities. *Annals of Statistics,* 33(4):1497–1537, 2005.

A. Beygelzimer, S. Dasgupta and J. Langford. Importance weighted active learning. In *Proceedings of the 26th International Conference on Machine Learning ICML 2009*, pages 49–56, Montreal, Canada, 2009.

R. Castro, R. Willett and R. Nowak. Faster rates in regression via active learning. In: *Advances in Neural Information Processing Systems 18 (NIPS 2005),* pages 179–186, 2005.

R. Castro and R. Nowak. Minimax bounds for active learning. *IEEE Transactions on Information Theory,* 54(5):2339–2353, 2008.

S. Dasgupta, D. Hsu and C. Monteleoni. A general agnostic active learning algorithm. In: *Advances in Neural Information Processing Systems 20 (NIPS 2007),* pages 353–360, 2007.

L. Devroye, L. Györfi and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

R.M. Dudley. *Uniform Central Limit Theorems*. Cambridge University Press, 1999.

E. Giné and V. Koltchinskii. Concentration inequalities and asymptotic results for ratio type empirical processes. *Annals of Probability,* 34(3):1143–1216, 2006.

E.J. Friedman. Active learning for smooth problems, In *Proceeding of the 22nd Annual Conference on Learning Theory, COLT 2009*, pages 343–352, Montreal, Canada, 2009.

S. Hanneke. Adaptive rates of convergence in active learning. In: *Proc. 22nd Annual Conference on Learning Theory, COLT 2009,* pages 353–364, Montreal, Canada, 2009a.

S. Hanneke. Rates of convergence in active learning. Preprint, 2009b.

V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory,* 47(5):1902–1914, 2001.

V. Koltchinskii. Local Rademacher complexities and oracle inequalities in risk minimization. *Annals of Statistics,* 34(6):2593–2656, 2006.

V. Koltchinskii. Oracle inequalities in empirical risk minimization and sparse recovery problems. Lecture Notes. *Ecole d'ete de Probabilités de Saint-Flour 2008,* Preprint, 2008.

V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. In: E. Giné, D. Mason and J. Wellner (Eds). *High Dimensional Probability II,* pages 443–459, Birkhäuser, Boston, 2000.

P. Massart and E. Nedelec. Risk bounds for statistical learning. *Annals of Statistics,* 34(5):2326–2366, 2006.

A. Tsybakov. Optimal aggregation in statistical learning. *Annals of Statistics,* 32(1):135–166, 2004.

A. van der Vaart and J. Wellner. *Weak Convergence and Empirical Processes*. Springer, New York, 1996.

Journal of Machine Learning Research 11 (2010) 2487-2531 Submitted 12/09; Revised 6/10; Published 9/10

Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data*

Miloš Radovanović RADACHA@DMI.UNS.AC.RS
Department of Mathematics and Informatics
University of Novi Sad
Trg D. Obradovića 4, 21000 Novi Sad, Serbia

Alexandros Nanopoulos NANOPOULOS@ISMLL.DE
Institute of Computer Science
University of Hildesheim
Marienburger Platz 22, D-31141 Hildesheim, Germany

Mirjana Ivanović MIRA@DMI.UNS.AC.RS
Department of Mathematics and Informatics
University of Novi Sad
Trg D. Obradovića 4, 21000 Novi Sad, Serbia

Editor: Ulrike von Luxburg

Abstract

Different aspects of the curse of dimensionality are known to present serious challenges to various machine-learning methods and tasks. This paper explores a new aspect of the dimensionality curse, referred to as *hubness*, that affects the distribution of k-occurrences: the number of times a point appears among the k nearest neighbors of other points in a data set. Through theoretical and empirical analysis involving synthetic and real data sets we show that under commonly used assumptions this distribution becomes considerably skewed as dimensionality increases, causing the emergence of *hubs*, that is, points with very high k-occurrences which effectively represent "popular" nearest neighbors. We examine the origins of this phenomenon, showing that it is an inherent property of data distributions in high-dimensional vector space, discuss its interaction with dimensionality reduction, and explore its influence on a wide range of machine-learning tasks directly or indirectly based on measuring distances, belonging to supervised, semi-supervised, and unsupervised learning families.

Keywords: nearest neighbors, curse of dimensionality, classification, semi-supervised learning, clustering

1. Introduction

The curse of dimensionality, a term originally introduced by Bellman (1961), is nowadays commonly used in many fields to refer to challenges posed by high dimensionality of data space. In the field of machine learning, affected methods and tasks include Bayesian modeling (Bishop, 2006), nearest-neighbor prediction (Hastie et al., 2009) and search (Korn et al., 2001), neural networks (Bishop, 1996), and many others. One aspect of the dimensionality curse is *distance concentration*, which denotes the tendency of distances between all pairs of points in high-dimensional data to become almost equal. Distance concentration and the meaningfulness of nearest neighbors in high

*. A preliminary version of this paper appeared in the Proceedings of the 26th International Conference on Machine Learning (Radovanović et al., 2009).

dimensions has been thoroughly explored (Beyer et al., 1999; Hinneburg et al., 2000; Aggarwal et al., 2001; François et al., 2007). The effect of the phenomenon on machine learning was demonstrated, for example, in studies of the behavior of kernels in the context of support vector machines, lazy learning, and radial basis function networks (Evangelista et al., 2006; François, 2007).

There exists another aspect of the curse of dimensionality that is related to nearest neighbors (NNs), which we will refer to as *hubness*. Let $D \subset \mathbb{R}^d$ be a set of d-dimensional points and $N_k(\mathbf{x})$ the number of *k-occurrences* of each point $\mathbf{x} \in D$, that is, the number of times $\mathbf{x}$ occurs among the k nearest neighbors of all other points in D, according to some distance measure. Under widely applicable conditions, as dimensionality increases, the distribution of N_k becomes considerably skewed to the right, resulting in the emergence of *hubs*, that is, points which appear in many more k-NN lists than other points, effectively making them "popular" nearest neighbors. Unlike distance concentration, hubness and its influence on machine learning have not been explored in depth. In this paper we study the causes and implications of this aspect of the dimensionality curse.

As will be described in Section 4, the phenomena of distance concentration and hubness are related, but distinct. Traditionally, distance concentration is studied through asymptotic behavior of norms, that is, distances to the origin, with increasing dimensionality. The obtained results trivially extend to reference points other than the origin, and to pairwise distances between all points. However, the asymptotic tendencies of distances of all points to different reference points do not necessarily occur at the same *speed*, which will be shown for normally distributed data by our main theoretical result outlined in Section 4.2, and given with full details in Section 5.1. The main consequence of the analysis, which is further discussed in Section 5.2 and supported by theoretical results by Newman et al. (1983) and Newman and Rinott (1985), is that the hubness phenomenon is an inherent property of data distributions in high-dimensional space under widely used assumptions, and not an artefact of a finite sample or specific properties of a particular data set.

The above result is relevant to machine learning because many families of ML algorithms, regardless of whether they are supervised, semi-supervised, or unsupervised, directly or indirectly make use of distances between data points (and, with them, k-NN graphs) in the process of building a model. Moreover, the hubness phenomenon recently started to be observed in application fields like music retrieval (Aucouturier and Pachet, 2007), speech recognition (Doddington et al., 1998), and fingerprint identification (Hicklin et al., 2005), where it is described as a problematic situation, but little or no insight is offered into the origins of the phenomenon. In this paper we present a unifying view of the hubness phenomenon through theoretical analysis of data distributions, and empirical investigation including numerous synthetic and real data sets, explaining the origins of the phenomenon and the mechanism through which hubs emerge, discussing the role of *antihubs* (points which appear in very few, if any, k-NN lists of other points), and studying the effects on common supervised, semi-supervised, and unsupervised machine-learning algorithms.

After discussing related work in the next section, we make the following contributions. First, we demonstrate the emergence of hubness on synthetic and real data in Section 3. The following section provides a comprehensive explanation of the origins of the phenomenon, through empirical and theoretical analysis of artificial data distributions, as well as observations on a large collection of real data sets, linking hubness with the *intrinsic* dimensionality of data. Section 5 presents the details of our main theoretical result which describes the mechanism through which hubs emerge as dimensionality increases, and provides discussion and further illustration of the behavior of nearest-neighbor relations in high dimensions, connecting our findings with existing theoretical results. The role of dimensionality reduction is discussed in Section 6, adding further support to the previ-

ously established link between intrinsic dimensionality and hubness, and demonstrating that dimensionality reduction may not constitute an easy mitigation of the phenomenon. Section 7 explores the impact of hubness on common supervised, semi-supervised, and unsupervised machine-learning algorithms, showing that the information provided by hubness can be used to significantly affect the success of the generated models. Finally, Section 8 concludes the paper, and provides guidelines for future work.

2. Related Work

The hubness phenomenon has been recently observed in several application areas involving sound and image data (Aucouturier and Pachet, 2007; Doddington et al., 1998; Hicklin et al., 2005). Also, Jebara et al. (2009) briefly mention hubness in the context of graph construction for semi-supervised learning. In addition, there have been attempts to avoid the influence of hubs in 1-NN time-series classification, apparently without clear awareness about the existence of the phenomenon (Islam et al., 2008), and to account for possible skewness of the distribution of N_1 in reverse nearest-neighbor search (Singh et al., 2003),[1] where $N_k(\mathbf{x})$ denotes the number of times point $\mathbf{x}$ occurs among the k nearest neighbors of all other points in the data set. None of the mentioned papers, however, successfully analyze the causes of hubness or generalize it to other applications. One recent work that makes the connection between hubness and dimensionality is the thesis by Berenzweig (2007), who observed the hubness phenomenon in the application area of music retrieval and identified high dimensionality as a cause, but did not provide practical or theoretical support that would explain the mechanism through which high dimensionality causes hubness in music data.

The distribution of N_1 has been explicitly studied in the applied probability community (Newman et al., 1983; Maloney, 1983; Newman and Rinott, 1985; Yao and Simons, 1996), and by mathematical psychologists (Tversky et al., 1983; Tversky and Hutchinson, 1986). In the vast majority of studied settings (for example, Poisson process, d-dimensional torus), coupled with Euclidean distance, it was shown that the distribution of N_1 converges to the Poisson distribution with mean 1, as the number of points n and dimensionality d go to infinity. Moreover, from the results by Yao and Simons (1996) it immediately follows that, in the Poisson process case, the distribution of N_k converges to the Poisson distribution with mean k, for any $k \geq 1$. All these results imply that no hubness is to be expected within the settings in question. On the other hand, in the case of continuous distributions with i.i.d. components, for the following specific order of limits it was shown that $\lim_{n\to\infty} \lim_{d\to\infty} \mathrm{Var}(N_1) = \infty$, while $\lim_{n\to\infty} \lim_{d\to\infty} N_1 = 0$, in distribution (Newman et al., 1983, p. 730, Theorem 7), with a more general result provided by Newman and Rinott (1985). According to the interpretation by Tversky et al. (1983), this suggests that if the number of dimensions is large relative to the number of points one may expect a small proportion of points to become hubs. However, the importance of this finding was downplayed to a certain extent (Tversky et al., 1983; Newman and Rinott, 1985), citing empirically observed slow convergence (Maloney, 1983), with the attention of the authors shifting more towards similarity measurements obtained directly from psychological and cognitive experiments (Tversky et al., 1983; Tversky and Hutchinson, 1986) that do not involve vector-space data. In Section 5.2 we will discuss the above results in more detail, as well as their relations with our theoretical and empirical findings.

It is worth noting that in ε-neighborhood graphs, that is, graphs where two points are connected if the *distance* between them is less than a given limit ε, the hubness phenomenon does not occur.

1. Reverse nearest-neighbor queries retrieve data points that have the query point $\mathbf{q}$ as their nearest neighbor.

Settings involving randomly generated points forming ε-neighborhood graphs are typically referred to as random geometric graphs, and are discussed in detail by Penrose (2003).

Concentration of distances, a phenomenon related to hubness, was studied for general distance measures (Beyer et al., 1999; Durrant and Kabán, 2009) and specifically for Minkowski and fractional distances (Demartines, 1994; Hinneburg et al., 2000; Aggarwal et al., 2001; François et al., 2007; François, 2007; Hsu and Chen, 2009). Concentration of cosine similarity was explored by Nanopoulos et al. (2009).

In our recent work (Radovanović et al., 2009), we performed an empirical analysis of hubness, its causes, and effects on techniques for classification, clustering, and information retrieval. In this paper we extend our findings with additional theoretical and empirical insight, offering a unified view of the origins and mechanics of the phenomenon, and its significance to various machine-learning applications.

3. The Hubness Phenomenon

In Section 1 we gave a simple set-based deterministic definition of N_k. To complement this definition and introduce N_k into a probabilistic setting that will also be considered in this paper, let $\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n$, be $n+1$ random vectors drawn from the same continuous probability distribution with support $\mathcal{S} \subseteq \mathbb{R}^d$, $d \in \{1,2,\ldots\}$, and let *dist* be a distance function defined on $\mathbb{R}^d$ (not necessarily a metric). Let functions $p_{i,k}$, where $i,k \in \{1,2,\ldots,n\}$, be defined as

$$p_{i,k}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \text{ is among the } k \text{ nearest neighbors of } \mathbf{x}_i, \text{ according to } dist, \\ 0, & \text{otherwise.} \end{cases}$$

In this setting, we define $N_k(\mathbf{x}) = \sum_{i=1}^{n} p_{i,k}(\mathbf{x})$, that is, $N_k(\mathbf{x})$ is the random number of vectors from $\mathbb{R}^d$ that have $\mathbf{x}$ included in their list of k nearest neighbors. In this section we will empirically demonstrate the emergence of hubness through increasing skewness of the distribution of N_k on synthetic (Section 3.1) and real data (Section 3.2), relating the increase of skewness with the dimensionality of data sets, and motivating the subsequent study into the origins of the phenomenon in Section 4.

3.1 A Motivating Example

We start with an illustrative experiment which demonstrates the changes in the distribution of N_k with varying dimensionality. Let us consider a random data set consisting of 10000 d-dimensional points, whose components are independently drawn from the uniform distribution in range $[0,1]$, and the following distance functions: Euclidean (l_2), fractional $l_{0.5}$ (proposed for high-dimensional data by Aggarwal et al., 2001), and cosine. Figure 1(a–c) shows the empirically observed distributions of N_k, with $k = 5$, for (a) $d = 3$, (b) $d = 20$, and (c) $d = 100$. In the same way, Figure 1(d–f) depicts the empirically observed N_k for points randomly drawn from the i.i.d. normal distribution.

For $d = 3$ the empirical distributions of N_5 for the three distance functions (Figure 1(a,d)) are consistent with the binomial distribution. This is expected by considering k-occurrences as node in-degrees in the k-nearest neighbor digraph. For randomly distributed points in low dimensions, the degree distributions of the digraphs closely resemble the degree distribution of the Erdős-Rényi (ER) random graph model, which is is binomial and Poisson in the limit (Erdős and Rényi, 1959).

As dimensionality increases, the observed distributions of N_5 depart from the random graph model and become more skewed to the right (Figure 1(b,c), and Figure 1(e,f) for l_2 and $l_{0.5}$).

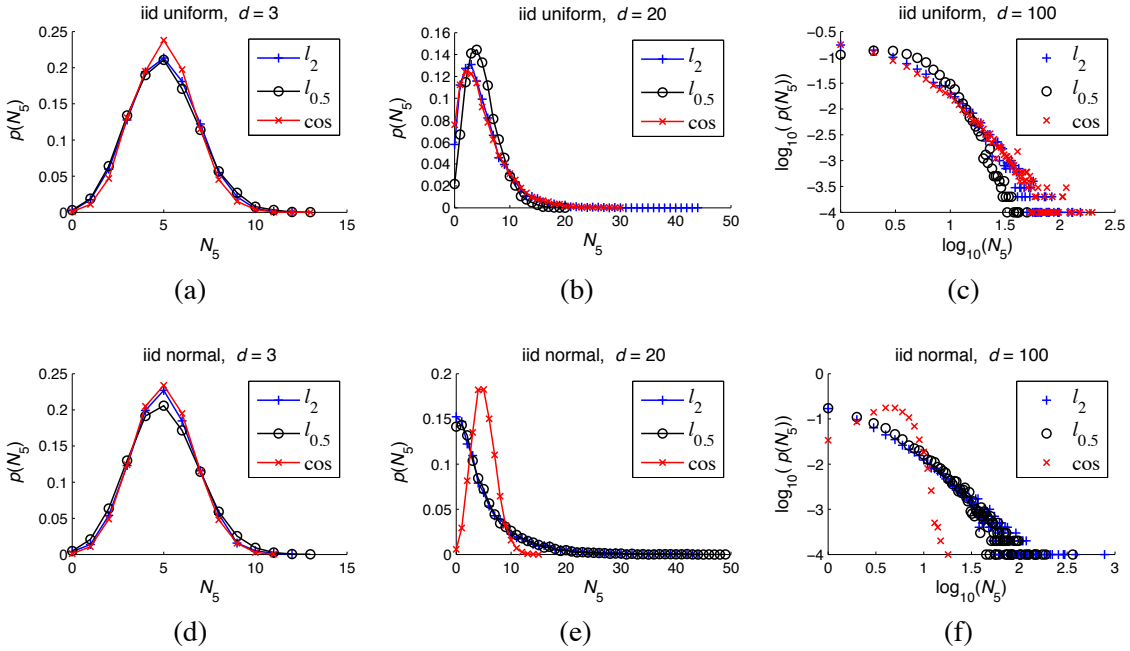


Figure 1: Empirical distribution of N_5 for Euclidean, $l_{0.5}$, and cosine distances on (a–c) i.i.d. uniform, and (d–f) i.i.d. normal random data sets with $n = 10000$ points and dimensionality (a,d) $d = 3$, (b,e) $d = 20$, and (c,f) $d = 100$ (log-log plot).

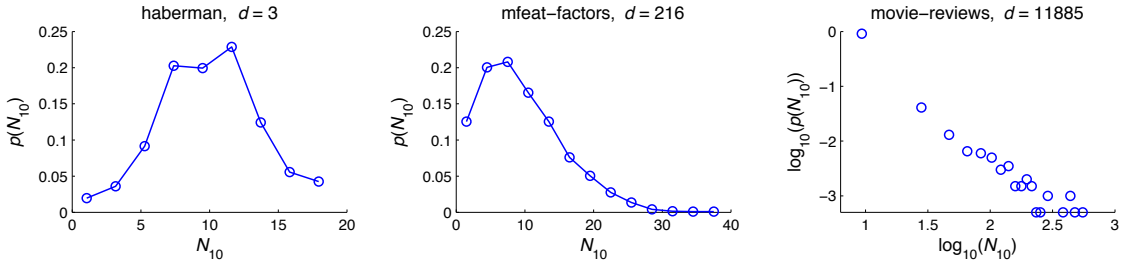
We verified this by being able to fit major right portions (that is, tails) of the observed distributions with the log-normal distribution, which is highly skewed.[2] We made similar observations with various k values (generally focusing on the common case $k \ll n$, where n is the number of points in a data set), distance measures (l_p-norm distances for both $p \geq 1$ and $0 < p < 1$, Bray-Curtis, normalized Euclidean, and Canberra), and data distributions. In virtually all these cases, skewness exists and produces hubs, that is, points with high k-occurrences. One exception visible in Figure 1 is the combination of cosine distance and normally distributed data. In most practical settings, however, such situations are not expected, and a thorough discussion of the necessary conditions for hubness to occur in high dimensions will be given in Section 5.2.

3.2 Hubness in Real Data

To illustrate the hubness phenomenon on real data, let us consider the empirical distribution of N_k ($k = 10$) for three real data sets, given in Figure 2. As in the previous section, a considerable increase in the skewness of the distributions can be observed with increasing dimensionality.

In all, we examined 50 real data sets from well known sources, belonging to three categories: UCI multidimensional data, gene expression microarray data, and textual data in the bag-of-words

2. Fits were supported by the χ^2 goodness-of-fit test at 0.05 significance level, where bins represent the number of observations of individual N_k values. These empirical distributions were compared with the expected output of a (discretized) log-normal distribution, making sure that counts in the bins do not fall below 5 by pooling the rightmost bins together.



(a) (b) (c)

Figure 2: Empirical distribution of N_{10} for three real data sets of different dimensionalities.

representation,[3] listed in Table 1. The table includes columns that describe data-set sources (2nd column), basic statistics (data transformation (3rd column): whether standardization was applied, or for textual data the used bag-of-words document representation; the number of points (n, 4th column); dimensionality (d, 5th column); the number of classes (7th column)), and the distance measure used (Euclidean or cosine, 8th column). We took care to ensure that the choice of distance measure and preprocessing (transformation) corresponds to a realistic scenario for the particular data set.

To characterize the asymmetry of N_k we use the standardized third moment of the distribution of k-occurrences,

$$S_{N_k} = \frac{\mathrm{E}(N_k - \mu_{N_k})^3}{\sigma_{N_k}^3},$$

where μ_{N_k} and σ_{N_k} are the mean and standard deviation of N_k, respectively. The corresponding (9th) column of Table 1, which shows the empirical $S_{N_{10}}$ values for the real data sets, indicates that the distributions of N_{10} for most examined data sets are skewed to the right.[4] The value of k is fixed at 10, but analogous observations can be made with other values of k.

It can be observed that some S_{N_k} values in Table 1 are quite high, indicating strong hubness in the corresponding data sets.[5] Moreover, computing the Spearman correlation between d and S_{N_k} over all 50 data sets reveals it to be strong (0.62), signifying that the relationship between dimensionality and hubness extends from synthetic to real data in general. On the other hand, careful scrutiny of the charts in Figure 2 and S_{N_k} values in Table 1 reveals that for real data the impact of dimensionality on hubness may not be as strong as could be expected after viewing hubness on synthetic data in Figure 1. Explanations for this observation will be given in the next section.

3. We used the movie review polarity data set v2.0 initially introduced by Pang and Lee (2004), while the computers and sports data sets were first used by Radovanović and Ivanović (2006). Preprocessing of all text data sets (except dexter, which is already preprocessed) involved stop-word removal and stemming using the Porter stemmer. Documents were transformed into the bag-of-words representation with word weights being either term frequencies (tf), or term frequencies multiplied by inverse document frequencies (tf-idf), with the choice based on independent experiments involving several classifiers. All term frequency vectors were normalized to average document length.
4. If $S_{N_k} = 0$ there is no skewness, positive (negative) values signify skewness to the right (left).
5. For comparison, sample skewness values for i.i.d. uniform data and Euclidean distance, shown in Figure 1(a–c), are 0.121, 1.541, and 5.445 for dimensionalities 3, 20, and 100, respectively. The values for i.i.d. normal data from Figure 1(d–f) are 0.118, 2.055, and 19.210.

Name	Src.	Trans.	n	d	d_{mle}	Cls.	Dist.	$S_{N_{10}}$	$S^S_{N_{10}}$	Clu.	$C^{N_{10}}_{\text{dm}}$	$C^{N_{10}}_{\text{cm}}$	$\widetilde{BN}_{10}$	$C^{N_{10}}_{BN_{10}}$	CAV
abalone	UCI	stan	4177	8	5.39	29	l_2	0.277	0.235	62	−0.047	−0.526	0.804	0.934	0.806
arcene	UCI	stan	100	10000	22.85	2	l_2	0.634	2.639	2	−0.559	−0.684	0.367	0.810	0.455
arrhythmia	UCI	stan	452	279	21.63	16	l_2	1.984	6.769	8	−0.867	−0.892	0.479	0.898	0.524
breast-w	UCI	stan	699	9	5.97	2	l_2	1.020	0.667	7	−0.062	−0.240	0.052	0.021	0.048
diabetes	UCI	stan	768	8	6.00	2	l_2	0.555	0.486	15	−0.479	−0.727	0.322	0.494	0.337
dorothea	UCI	none	800	100000	201.11	2	cos	2.355	1.016	19	−0.632	−0.672	0.108	0.236	0.092
echocardiogram	UCI	stan	131	7	4.92	2	l_2	0.735	0.438	5	−0.722	−0.811	0.372	0.623	0.337
ecoli	UCI	stan	336	7	4.13	8	l_2	0.116	0.208	8	−0.396	−0.792	0.223	0.245	0.193
gisette	UCI	none	6000	5000	149.35	2	cos	1.967	4.671	76	−0.667	−0.854	0.045	0.367	0.241
glass	UCI	stan	214	9	4.37	7	l_2	0.154	0.853	11	−0.430	−0.622	0.414	0.542	0.462
haberman	UCI	stan	306	3	2.89	2	l_2	0.087	−0.316	11	−0.330	−0.573	0.348	0.305	0.360
ionosphere	UCI	stan	351	34	13.57	2	l_2	1.717	2.051	18	−0.639	−0.832	0.185	0.464	0.259
iris	UCI	stan	150	4	2.96	3	l_2	0.126	−0.068	4	−0.275	−0.681	0.087	0.127	0.147
isolet1	UCI	stan	1560	617	13.72	26	l_2	1.125	6.483	38	−0.306	−0.760	0.283	0.463	0.352
mfeat-factors	UCI	stan	2000	216	8.47	10	l_2	0.826	5.493	44	−0.113	−0.688	0.063	0.001	0.145
mfeat-fourier	UCI	stan	2000	76	11.48	10	l_2	1.277	4.001	44	−0.350	−0.596	0.272	0.436	0.415
mfeat-karhunen	UCI	stan	2000	64	11.82	10	l_2	1.250	8.671	40	−0.436	−0.788	0.098	0.325	0.205
mfeat-morph	UCI	stan	2000	6	3.22	10	l_2	−0.153	0.010	44	−0.039	−0.424	0.324	0.306	0.397
mfeat-pixel	UCI	stan	2000	240	11.83	10	l_2	1.035	3.125	44	−0.210	−0.738	0.049	0.085	0.107
mfeat-zernike	UCI	stan	2000	47	7.66	10	l_2	0.933	3.389	44	−0.185	−0.657	0.235	0.252	0.400
musk1	UCI	stan	476	166	6.74	2	l_2	1.327	3.845	17	−0.376	−0.752	0.237	0.621	0.474
optdigits	UCI	stan	5620	64	9.62	10	l_2	1.095	3.789	74	−0.223	−0.601	0.044	0.097	0.168
ozone-eighthr	UCI	stan	2534	72	12.92	2	l_2	2.251	4.443	49	−0.216	−0.655	0.086	0.300	0.138
ozone-onehr	UCI	stan	2536	72	12.92	2	l_2	2.260	5.798	49	−0.215	−0.651	0.046	0.238	0.070
page-blocks	UCI	stan	5473	10	3.73	5	l_2	−0.014	0.470	72	−0.063	−0.289	0.049	−0.046	0.068
parkinsons	UCI	stan	195	22	4.36	2	l_2	0.729	1.964	8	−0.414	−0.649	0.166	0.321	0.256
pendigits	UCI	stan	10992	16	5.93	10	l_2	0.435	0.982	104	−0.062	−0.513	0.014	−0.030	0.156
segment	UCI	stan	2310	19	3.93	7	l_2	0.313	1.111	48	−0.077	−0.453	0.089	0.074	0.332
sonar	UCI	stan	208	60	9.67	2	l_2	1.354	3.053	8	−0.550	−0.771	0.286	0.632	0.461
spambase	UCI	stan	4601	57	11.45	2	l_2	1.916	2.292	49	−0.376	−0.448	0.139	0.401	0.271
spectf	UCI	stan	267	44	13.83	2	l_2	1.895	2.098	11	−0.616	−0.729	0.300	0.595	0.366
spectrometer	UCI	stan	531	100	8.04	10	l_2	0.591	3.123	17	−0.269	−0.670	0.200	0.225	0.242
vehicle	UCI	stan	846	18	5.61	4	l_2	0.603	1.625	25	−0.162	−0.643	0.358	0.435	0.586
vowel	UCI	stan	990	10	2.39	11	l_2	0.766	0.935	27	−0.252	−0.605	0.313	0.691	0.598
wdbc	UCI	stan	569	30	8.26	2	l_2	0.815	3.101	16	−0.449	−0.708	0.065	0.170	0.129
wine	UCI	stan	178	13	6.69	3	l_2	0.630	1.319	3	−0.589	−0.874	0.076	0.182	0.084
wpbc	UCI	stan	198	33	8.69	2	l_2	0.863	2.603	6	−0.688	−0.878	0.340	0.675	0.360
yeast	UCI	stan	1484	8	5.42	10	l_2	0.228	0.105	34	−0.421	−0.715	0.527	0.650	0.570
AMLALL	KR	none	72	7129	31.92	2	l_2	1.166	1.578	2	−0.868	−0.927	0.171	0.635	0.098
colonTumor	KR	none	62	2000	11.22	2	l_2	1.055	1.869	3	−0.815	−0.781	0.305	0.779	0.359
DLBCL	KR	none	47	4026	16.11	2	l_2	1.007	1.531	2	−0.942	−0.947	0.338	0.895	0.375
lungCancer	KR	none	181	12533	59.66	2	l_2	1.248	3.073	6	−0.537	−0.673	0.052	0.262	0.136
MLL	KR	none	72	12582	28.42	3	l_2	0.697	1.802	2	−0.794	−0.924	0.211	0.533	0.148
ovarian-61902	KR	none	253	15154	9.58	2	l_2	0.760	3.771	10	−0.559	−0.773	0.164	0.467	0.399
computers	dmoz	tf	697	1168	190.33	2	cos	2.061	2.267	26	−0.566	−0.731	0.312	0.699	0.415
dexter	UCI	none	300	20000	160.78	2	cos	3.977	4.639	13	−0.760	−0.781	0.301	0.688	0.423
mini-newsgroups	UCI	tf-idf	1999	7827	3226.43	20	cos	1.980	1.765	44	−0.422	−0.704	0.524	0.701	0.526
movie-reviews	PaBo	tf	2000	11885	54.95	2	cos	8.796	7.247	44	−0.604	−0.739	0.398	0.790	0.481
reuters-transcribed	UCI	tf-idf	201	3029	234.68	10	cos	1.165	1.693	3	−0.781	−0.763	0.642	0.871	0.595
sports	dmoz	tf	752	1185	250.24	2	cos	1.629	2.543	27	−0.584	−0.736	0.260	0.604	0.373

Table 1: Real data sets. Data sources are the University of California, Irvine (UCI) Machine Learning Repository, Kent Ridge (KR) Bio-Medical Data Set Repository, dmoz Open Directory, and www.cs.cornell.edu/People/pabo/movie-review-data/ (PaBo).

4. The Origins of Hubness

This section moves on to exploring the causes of hubness and the mechanisms through which hubs emerge. Section 4.1 investigates the relationship between the position of a point in data space and hubness. Next, Section 4.2 explains the mechanism through which hubs emerge as points in high-dimensional space that become closer to other points than their low-dimensional counterparts, outlining our main theoretical result. The emergence of hubness in real data is studied in Section 4.3, while Section 4.4 discusses hubs and their opposites—antihubs—and the relationships between hubs, antihubs, and different notions of outliers.

4.1 The Position of Hubs

Let us consider again the i.i.d. uniform and i.i.d. normal random data examined in the previous section. We will demonstrate that the position of a point in data space has a significant effect on its k-occurrences value, by observing the sample mean of the data distribution as a point of reference. Figure 3 plots, for each point $\mathbf{x}$, its $N_5(\mathbf{x})$ against its Euclidean distance from the empirical data mean, for $d = 3,\ 20,\ 100$. As dimensionality increases, stronger correlation emerges, implying that points closer to the mean tend to become hubs. We made analogous observations with other values of k, and combinations of data distributions and distance measures for which hubness occurs. It is important to note that proximity to one global data-set mean correlates with hubness in high dimensions when the underlying data distribution is *unimodal*. For multimodal data distributions, for example those obtained through a mixture of unimodal distributions, hubs tend to appear close to the means of individual component distributions of the mixture. In the discussion that follows in Section 4.2 we will assume a unimodal data distribution, and defer the analysis of multimodal distributions until Section 4.3, which studies real data.

4.2 Mechanisms Behind Hubness

Although one may expect that some random points are closer to the data-set mean than others, in order to explain the mechanism behind hub formation we need to (1) understand the geometrical and distributional setting in which some points tend to be closer to the mean than others, and then (2) understand why such points become hubs in higher dimensions.[6]

Hubness appears to be related to the phenomenon of distance concentration, which is usually expressed as the ratio between some measure of spread (for example, standard deviation) and some measure of magnitude (for example, the mean) of distances of all points in a data set to some arbitrary reference point (Aggarwal et al., 2001; François et al., 2007). If this ratio converges to 0 as dimensionality goes to infinity, it is said that the distances concentrate. Based on existing theoretical results discussing distance concentration (Beyer et al., 1999; Aggarwal et al., 2001), high-dimensional points are approximately lying on a hypersphere centered at the data-set mean. Moreover, the results by Demartines (1994) and François et al. (2007) specify that the distribution of distances to the data-set mean has a non-negligible variance for any finite d.[7] Hence, the existence of a non-negligible number of points closer to the data-set mean is *expected* in high dimensions.

6. We will assume that random points originate from a unimodal data distribution. In the multimodal case, it can be said that the observations which follow are applicable around one of the "peaks" in the pdf of the data distribution.

7. These results apply to l_p-norm distances, but our numerical simulations suggest that other distance functions mentioned in Section 3.1 behave similarly. Moreover, any point can be used as a reference instead of the data mean, but we observe the data mean since it plays a special role with respect to hubness.

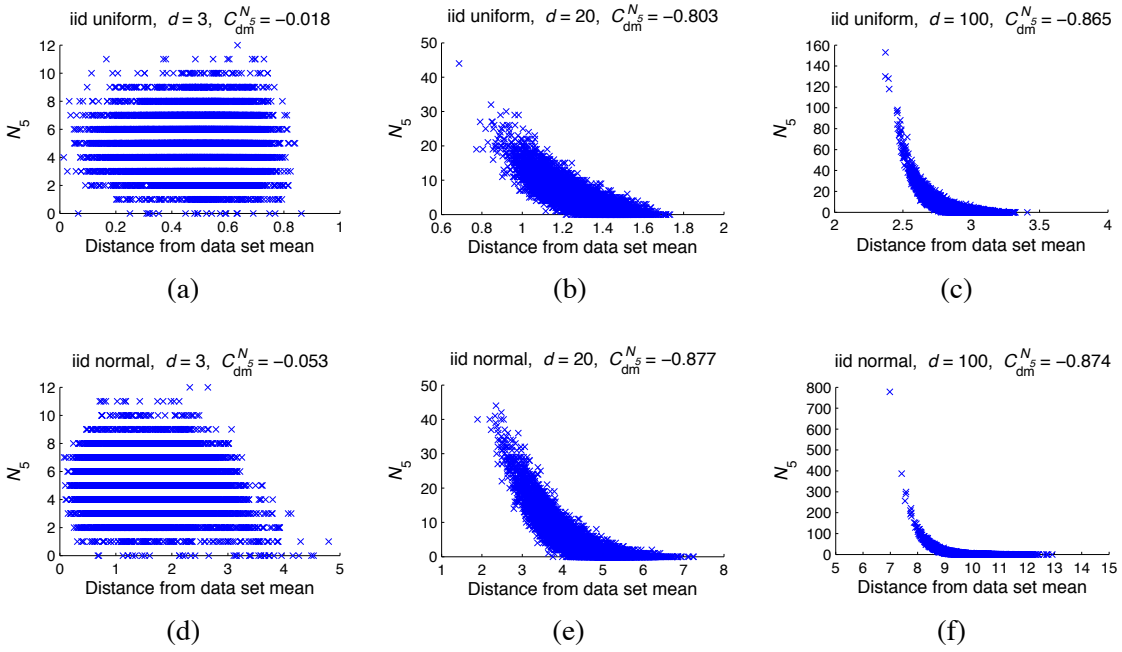


Figure 3: Scatter plots and Spearman correlation of $N_5(\mathbf{x})$ against the Euclidean distance of point $\mathbf{x}$ to the sample data-set mean for (a–c) i.i.d. uniform and (d–f) i.i.d. normal random data sets with (a, d) $d = 3$, (b, e) $d = 20$, and (c, f) $d = 100$.

To illustrate the above discussion, Figure 4 depicts, for i.i.d. normal data, the distribution of Euclidean distances of all points to the true data mean (the origin) for several d values. By definition, the distribution of distances is actually the Chi distribution with d degrees of freedom (as the square root of the sum of squares of i.i.d. normal variables, Johnson et al., 1994).[8] In this setting, distance concentration refers to the fact that the standard deviation of distance distributions is asymptotically constant with respect to increasing d, while the means of the distance distributions asymptotically behave like $\sqrt{d}$ (a direct consequence of the results by François et al., 2007, discussed further in Section 5.1). On the other hand, for l_p-norm distances with $p > 2$, the standard deviation would tend to 0 (François et al., 2007). However, for any finite d, existing variation in the values of random coordinates causes some points to become closer to the distribution mean than others. This happens despite the fact that all distance values, in general, may be increasing together with d.

To understand why points closer to the data mean become hubs in high dimensions, let us consider the following example. We observe, within the i.i.d. normal setting, two points drawn from the data, but at specific positions with respect to the origin: point $\mathbf{b}_d$ which is at the expected distance from the origin, and point $\mathbf{a}_d$ which is two standard deviations closer. In light of the above, the distances of $\mathbf{a}_d$ and $\mathbf{b}_d$ from the origin change with increasing d, and it could be said that different $\mathbf{a}_d$-s (and $\mathbf{b}_d$-s) occupy analogous positions in the data spaces, with respect to changing d. The distances of $\mathbf{a}_d$ (and $\mathbf{b}_d$) to all other points, again following directly from the definition (Oberto and Pennecchi, 2006), are distributed according to *noncentral* Chi distributions with d degrees of

8. For this reason, in Figure 4 we plot the known pdf, not the empirically obtained distribution.

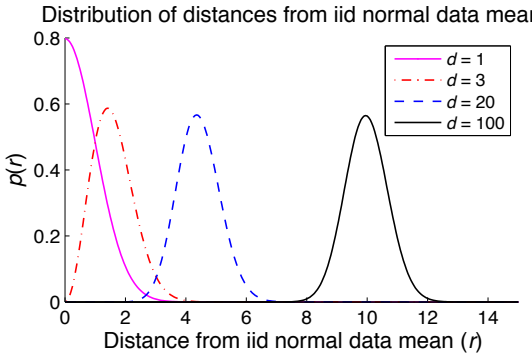
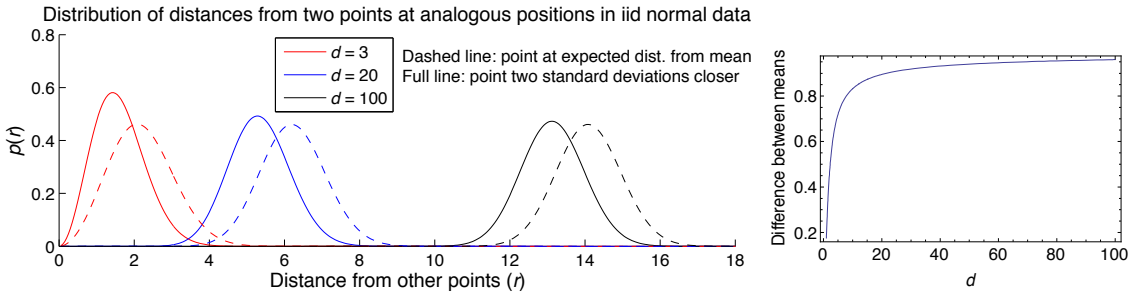


Figure 4: Probability density function of observing a point at distance r from the mean of a multivariate d-dimensional normal distribution, for $d = 1,\ 3,\ 20,\ 100$.



(a) (b)

Figure 5: (a) Distribution of distances to other points from i.i.d. normal random data for a point at the expected distance from the origin (dashed line), and a point two standard deviations closer (full line). (b) Difference between the means of the two distributions, with respect to increasing d.

freedom and noncentrality parameter λ equaling the distance of $\mathbf{a}_d$ ($\mathbf{b}_d$) to the origin. Figure 5(a) plots the probability density functions of these distributions for several values of d. It can be seen that, as d increases, the distance distributions for $\mathbf{a}_d$ and $\mathbf{b}_d$ move away from each other. This tendency is depicted more clearly in Figure 5(b) which plots the difference between the means of the two distributions with respect to d.

It is known, and expected, for points that are closer to the mean of the data distribution to be closer, on average, to all other points, for any value of d. However, the above analysis indicates that this tendency is amplified by high dimensionality, making points that reside in the proximity of the data mean become closer (in relative terms) to all other points than their low-dimensional analogues are. This tendency causes high-dimensional points that are closer to the mean to have increased inclusion probability into k-NN lists of other points, even for small values of k. We will discuss this relationship further in Section 5.2.

In terms of the notion of node *centrality* typically used in network analysis (Scott, 2000), the above discussion indicates that high dimensionality amplifies what we will call the *spatial centrality*

of a point (by increasing its proximity to other points), which, in turn, affects the *degree centrality* of the corresponding node in the k-NN graph (by increasing its degree, that is, N_k). Other notions of node centrality, and the structure of the k-NN graph in general, will be studied in more detail in Section 5.2.1. The rest of this section will focus on describing the mechanism of the observed spatial centrality amplification.

In the preceding discussion we selected two points from i.i.d. normal data with specific distances from the origin expressed in terms of expected distance and deviation from it, and tracked the analogues of the two points for increasing values of dimensionality d. Generally, we can express the distances of the two points to the origin in terms of "offsets" from the expected distance measured by standard deviation, which in the case of i.i.d. normal random data would be $\lambda_{d,1} = \mu_{\chi(d)} + c_1\sigma_{\chi(d)}$ and $\lambda_{d,2} = \mu_{\chi(d)} + c_2\sigma_{\chi(d)}$, where $\lambda_{d,1}$ and $\lambda_{d,2}$ are the distances of the first and second point to the origin, $\mu_{\chi(d)}$ and $\sigma_{\chi(d)}$ are the mean and standard deviation of the Chi distribution with d degrees of freedom, and c_1 and c_2 are selected constants (the offsets). In the preceding example involving points $\mathbf{a}_d$ and $\mathbf{b}_d$, we set $c_1 = -2$ and $c_2 = 0$, respectively. However, analogous behavior can be observed with arbitrary two points whose distance to the data mean is below the expected distance, that is, for $c_1, c_2 \leq 0$. We describe this behavior by introducing the following notation: $\Delta\mu_d(\lambda_{d,1}, \lambda_{d,2}) = |\mu_{\chi(d,\lambda_{d,2})} - \mu_{\chi(d,\lambda_{d,1})}|$, where $\mu_{\chi(d,\lambda_{d,i})}$ is the mean of the noncentral Chi distribution with d degrees of freedom and noncentrality parameter $\lambda_{d,i}$ ($i \in \{1,2\}$). In the following theorem, which we prove in Section 5.1, we show that $\Delta\mu_d(\lambda_{d,1}, \lambda_{d,2})$ increases with increasing values of d.

Theorem 1 *Let $\lambda_{d,1} = \mu_{\chi(d)} + c_1\sigma_{\chi(d)}$ and $\lambda_{d,2} = \mu_{\chi(d)} + c_2\sigma_{\chi(d)}$, where $d \in \mathbb{N}^+$, $c_1, c_2 \leq 0$, $c_1 < c_2$, and $\mu_{\chi(d)}$ and $\sigma_{\chi(d)}$ are the mean and standard deviation of the Chi distribution with d degrees of freedom, respectively. Define*

$$\Delta\mu_d(\lambda_{d,1}, \lambda_{d,2}) = \mu_{\chi(d,\lambda_{d,2})} - \mu_{\chi(d,\lambda_{d,1})}\,,$$

where $\mu_{\chi(d,\lambda_{d,i})}$ is the mean of the noncentral Chi distribution with d degrees of freedom and noncentrality parameter $\lambda_{d,i}$ ($i \in \{1,2\}$).

There exists $d_0 \in \mathbb{N}$ such that for every $d > d_0$,

$$\Delta\mu_d(\lambda_{d,1}, \lambda_{d,2}) > 0\,,$$

and

$$\Delta\mu_{d+2}(\lambda_{d+2,1}, \lambda_{d+2,2}) > \Delta\mu_d(\lambda_{d,1}, \lambda_{d,2})\,. \tag{1}$$

The main statement of the theorem is given by Equation 1, which expresses the tendency of the difference between the means of the two distance distributions to increase with increasing dimensionality d. It is important to note that this tendency is obtained through analysis of *distributions* of data and distances, implying that the behavior is an inherent property of data distributions in high-dimensional space, rather than an artefact of other factors, such as finite sample size, etc. Through simulation involving randomly generated points we verified the behavior for i.i.d. normal data by replicating very closely the chart shown in Figure 5(b). Furthermore, simulations suggest that the same behavior emerges in i.i.d. uniform data,[9] as well as numerous other unimodal random data distributions, producing charts of the same shape as in Figure 5(b). Real data, on the other hand,

9. The uniform cube setting will be discussed in more detail in Section 5.2, in the context of results from related work (Newman and Rinott, 1985).

tends to be clustered, and can be viewed as originating from a *mixture* of distributions resulting in a multimodal distribution of data. In this case, the behavior described by Theorem 1, and illustrated in Figure 5(b), is manifested primarily on the individual component distributions of the mixture, that is, on clusters of data points. The next section takes a closer look at the hubness phenomenon in real data sets.

4.3 Hubness in Real Data

Results describing the origins of hubness given in the previous sections were obtained by examining data sets that follow specific distributions and generated as i.i.d. samples from these distributions. To extend these results to real data, we need to take into account two additional factors: (1) real data sets usually contain dependent attributes, and (2) real data sets are usually clustered, that is, points are organized into groups produced by a mixture of distributions instead of originating from a single (unimodal) distribution.

To examine the first factor (dependent attributes), we adopt the approach of François et al. (2007) used in the context of distance concentration. For each data set we randomly permute the elements within every attribute. This way, attributes preserve their individual distributions, but the dependencies between them are lost and the *intrinsic dimensionality* of data sets increases, becoming equal to their embedding dimensionality d (François et al., 2007). In Table 1 (10th column) we give the empirical skewness, denoted as $S^S_{N_k}$, of the shuffled data. For the vast majority of high-dimensional data sets, $S^S_{N_k}$ is considerably higher than S_{N_k}, indicating that hubness actually depends on the intrinsic rather than embedding dimensionality. This provides an explanation for the apparent weaker influence of d on hubness in real data than in synthetic data sets, which was observed in Section 3.2.

To examine the second factor (many groups), for every data set we measured: (i) the Spearman correlation, denoted as $C^{N_{10}}_{\mathrm{dm}}$ (12th column), of the observed N_k and distance from the data-set mean, and (ii) the correlation, denoted as $C^{N_{10}}_{\mathrm{cm}}$ (13th column), of the observed N_k and distance to the closest group mean. Groups are determined with K-means clustering, where the number of clusters for each data set, given in column 11 of Table 1, was determined by exhaustive search of values between 2 and $\lfloor\sqrt{n}\rfloor$, to maximize $C^{N_{10}}_{\mathrm{cm}}$.[10] In most cases, $C^{N_{10}}_{\mathrm{cm}}$ is considerably stronger than $C^{N_{10}}_{\mathrm{dm}}$. Consequently, in real data, hubs tend to be closer than other points to their respective cluster centers (which we verified by examining the individual scatter plots).

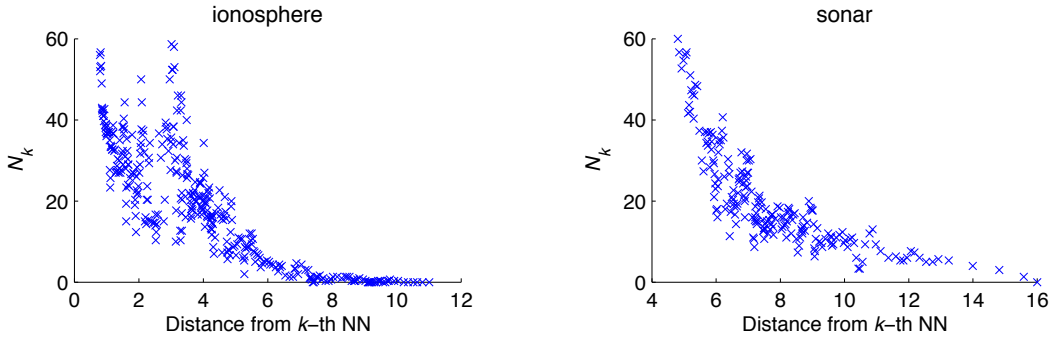
To further support the above findings, we include the 6th column (d_{mle}) to Table 1, corresponding to intrinsic dimensionality measured by the maximum likelihood estimator (Levina and Bickel, 2005). Next, we compute Spearman correlations between various measurements from Table 1 over all 50 examined data sets, given in Table 2. The observed skewness of N_k, besides being strongly correlated with d, is even more strongly correlated with the intrinsic dimensionality d_{mle}. Moreover, intrinsic dimensionality positively affects the correlations between N_k and the distance to the data-set mean / closest cluster mean, implying that in higher (intrinsic) dimensions the positions of hubs become increasingly localized to the proximity of centers.

Section 6, which discusses the interaction of hubness with dimensionality reduction, will provide even more support to the observation that hubness depends on intrinsic, rather than embedding dimensionality.

10. We report averages of $C^{N_{10}}_{\mathrm{cm}}$ over 10 runs of K-means clustering with different random seeding, in order to reduce the effects of chance.

	d	d_{mle}	$S_{N_{10}}$	$C_{\text{dm}}^{N_{10}}$	$C_{\text{cm}}^{N_{10}}$	$\widetilde{BN}_{10}$	$C_{BN_{10}}^{N_{10}}$
d_{mle}	0.87						
$S_{N_{10}}$	0.62	0.80					
$C_{\text{dm}}^{N_{10}}$	−0.52	−0.60	−0.42				
$C_{\text{cm}}^{N_{10}}$	−0.43	−0.48	−0.31	0.82			
$\widetilde{BN}_{10}$	−0.05	0.03	−0.08	−0.32	−0.18		
$C_{BN_{10}}^{N_{10}}$	0.32	0.39	0.29	−0.61	−0.46	0.82	
CAV	0.03	0.03	0.03	−0.14	−0.05	0.85	0.76

Table 2: Spearman correlations over 50 real data sets.



(a) (b)

Figure 6: Correlation between low N_k and outlier score ($k = 20$).

4.4 Hubs and Outliers

The non-negligible variance of the distribution of distances to the data mean described in Section 4.2 has an additional "side": we also expect points farther from the mean and, therefore, with much lower observed N_k than the rest.[11] Such points correspond to the bottom-right parts of Figure 3(b, c, e, f), and will be referred to as *antihubs*. Since antihubs are far away from all other points, in high dimensions they can be regarded as distance-based *outliers* (Tan et al., 2005).

To further support the connection between antihubs and distance-based outliers, let us consider a common outlier score of a point as the distance from its kth nearest neighbor (Tan et al., 2005). Low N_k values and high outlier scores are correlated as exemplified in Figure 6(a, b) (in their lower-right parts) for two data sets from Table 1.

Next, let us recall the i.i.d. normal random data setting, and the probability density function corresponding to observing a point at a specified distance from the mean, plotted in Figure 4. An analogous chart for real data is given in Figure 7, which shows the empirical distributions of distances from the closest cluster mean for three real data sets, as described in Section 4.3. In both figures it can be seen that in low dimensions the probability of observing a point near a center is quite high, while as dimensionality increases it becomes close to zero. If we now consider a *probabilistic* definition of an outlier as a point with a low probability of occurrence (Tan et al., 2005), in high

11. Assuming the presence of hubs, the existence of points with low N_k is implied by the constant-sum property of N_k: for any data set D, $\sum_{\mathbf{x}\in D} N_k(\mathbf{x}) = k|D|$ (Aucouturier and Pachet, 2007).

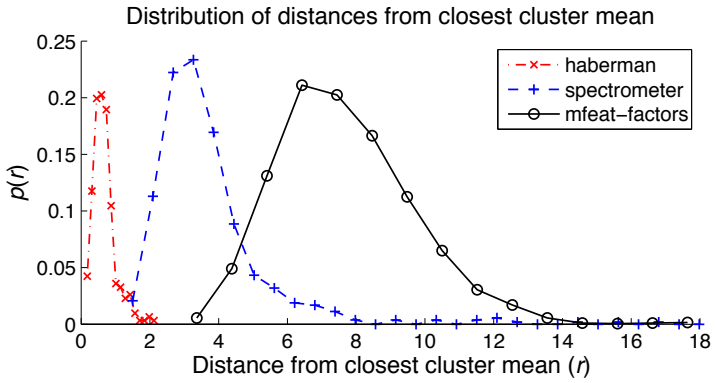


Figure 7: Probability density function of observing a point at distance r from the closest cluster mean for three real data sets.

dimensions hubs actually *are* outliers, as points closer to the distribution (or cluster) mean than the majority of other points. Therefore, somewhat counterintuitively, it can be said that hubs are points that reside in low-density regions of the high-dimensional data space, and are at the same time close to many other points. On the other hand, distance-based outliers correspond to probabilistic outliers that are farther away from the center(s). It follows that the definitions of distance-based and probabilistic outliers significantly diverge from one another in high dimensions, with distance-based outliers only partially corresponding to probabilistic outliers. To prevent confusion in the rest of the paper, we shall continue to refer to "hubs" and "outliers" in the distance-based sense. Outliers will be analyzed further in Section 7.3.2.

5. Proofs and Discussion

This section is predominantly devoted to the theoretical aspects of the behavior of distances in high-dimensional space and the hubness phenomenon. Section 5.1 provides a step-by-step proof of Theorem 1 which was introduced in Section 4.2, while Section 5.2 discusses additional aspects of the phenomenon in the context of the geometry of high-dimensional spaces and the properties of data distributions which occupy them.

5.1 Proof of Theorem 1

In this section we analyze the behavior of distances that provides the mechanism for the formation of hubs, introduced in Section 4.2, culminating with the proof of Theorem 1. Section 5.1.1 reviews distance concentration results by François et al. (2007), while Section 5.1.2 discusses distance distributions in i.i.d. normal random data and extended interpretations of the results by François et al. (2007) in this setting. The notion of asymptotic equivalence that will be used in the proof of Theorem 1 is the subject of Section 5.1.3. The expectation of the noncentral Chi distribution, which is a key feature in the analysis of distance distributions in i.i.d. normal random data, is defined in Section 5.1.4, together with the generalized Laguerre function on which it relies. Properties of the generalized Laguerre function that will be used in the proof of Theorem 1 are presented in Section 5.1.5. Finally, the proof of Theorem 1 is given in Section 5.1.6.

5.1.1 DISTANCE CONCENTRATION RESULTS

We begin by reviewing the main results of François et al. (2007) regarding distance concentration. Let $\mathbf{X}_d = (X_1, X_2, \ldots, X_d)$ be a random d-dimensional variable with i.i.d. components: $X_i \sim \mathcal{F}$, and let $\|\mathbf{X}_d\|$ denote its Euclidean norm. For random variables $|X_i|^2$, let $\mu_{|\mathcal{F}|^2}$ and $\sigma^2_{|\mathcal{F}|^2}$ signify their mean and variance, respectively. François et al. (2007) prove the following two lemmas.[12]

Lemma 2 (François et al., 2007, Equation 17, adapted)

$$\lim_{d\to\infty} \frac{\mathrm{E}(\|\mathbf{X}_d\|)}{\sqrt{d}} = \mu_{|\mathcal{F}|^2}\,.$$

Lemma 3 (François et al., 2007, Equation 21, adapted)

$$\lim_{d\to\infty} \mathrm{Var}(\|\mathbf{X}_d\|) = \frac{\sigma^2_{|\mathcal{F}|^2}}{4\mu_{|\mathcal{F}|^2}}\,.$$

The above lemmas imply that, for i.i.d. random data, the expectation of the distribution of Euclidean distances to the origin (Euclidean norms) asymptotically behaves like $\sqrt{d}$, while the standard deviation is asymptotically constant. From now on, we will denote the mean and variance of random variables that are distributed according to some distribution $\mathcal{F}$ by $\mu_{\mathcal{F}}$ and $\sigma^2_{\mathcal{F}}$, respectively.

5.1.2 DISTANCES IN I.I.D. NORMAL DATA

We now observe more closely the behavior of distances in i.i.d. normal random data. Let $\mathbf{Z}_d = (Z_1, Z_2, \ldots, Z_d)$ be a random d-dimensional vector whose components independently follow the standard normal distribution: $Z_i \sim \mathcal{N}(0;1)$, for every $i \in \{1, 2, \ldots, d\}$. Then, by definition, random variable $\|\mathbf{Z}_d\|$ follows the Chi distribution with d degrees of freedom: $\|\mathbf{Z}_d\| \sim \chi(d)$. In other words, $\chi(d)$ is the distribution of Euclidean distances of vectors drawn from $\mathbf{Z}_d$ to the origin. If one were to fix another reference vector $\mathbf{x}_d$ instead of the origin, the distribution of distances of vectors drawn from $\mathbf{Z}_d$ to $\mathbf{x}_d$ would be completely determined by $\|\mathbf{x}_d\|$ since, again by definition, random variable $\|\mathbf{Z}_d - \mathbf{x}_d\|$ follows the noncentral Chi distribution with d degrees of freedom and noncentrality parameter $\lambda = \|\mathbf{x}_d\|$: $\|\mathbf{Z}_d - \mathbf{x}_d\| \sim \chi(d, \|\mathbf{x}_d\|)$.

In light of the above, let us observe two points, $\mathbf{x}_{d,1}$ and $\mathbf{x}_{d,2}$, drawn from $\mathbf{Z}_d$, and express their distances from the origin in terms of offsets from the expected distance, with the offsets described using standard deviations: $\|\mathbf{x}_{d,1}\| = \mu_{\chi(d)} + c_1\sigma_{\chi(d)}$ and $\|\mathbf{x}_{d,2}\| = \mu_{\chi(d)} + c_2\sigma_{\chi(d)}$, where $c_1, c_2 \leq 0$. We will assume $c_1 < c_2$, that is, $\mathbf{x}_{d,1}$ is closer to the data distribution mean (the origin) than $\mathbf{x}_{d,2}$. By treating c_1 and c_2 as constants, and varying d, we observe analogues of two points in spaces of different dimensionalities (roughly speaking, points $\mathbf{x}_{d,1}$ have identical "probability" of occurrence at the specified distance from the origin for every d, and the same holds for $\mathbf{x}_{d,2}$). Let $\lambda_{d,1} = \|\mathbf{x}_{d,1}\|$ and $\lambda_{d,2} = \|\mathbf{x}_{d,2}\|$. Then, the distributions of distances of points $\mathbf{x}_{d,1}$ and $\mathbf{x}_{d,2}$ to all points from the data distribution $\mathbf{Z}_d$ (that is, the distributions of random variables $\|\mathbf{Z}_d - \mathbf{x}_{d,1}\|$ and $\|\mathbf{Z}_d - \mathbf{x}_{d,2}\|$) are noncentral Chi distributions $\chi(d, \lambda_{d,1})$ and $\chi(d, \lambda_{d,2})$, respectively. We study the behavior of these two distributions with increasing values of d.

Lemmas 2 and 3, taking $\mathcal{F}$ to be the standard normal distribution $\mathcal{N}(0;1)$, and translating the space so that $\mathbf{x}_{d,1}$ or $\mathbf{x}_{d,2}$ become the origin, imply that both $\mu_{\chi(d,\lambda_{d,1})}$ and $\mu_{\chi(d,\lambda_{d,2})}$ asymptotically

12. François et al. (2007) provide a more general result for l_p norms with arbitrary $p > 0$.

behave like $\sqrt{d}$ as $d \to \infty$, while $\sigma^2_{\chi(d,\lambda_{d,1})}$ and $\sigma^2_{\chi(d,\lambda_{d,2})}$ are both asymptotically constant.[13] However, for $\mathbf{x}_{d,1}$ and $\mathbf{x}_{d,2}$ placed at different distances form the origin ($\lambda_{d,1} \neq \lambda_{d,2}$, that is, $c_1 \neq c_2$), these asymptotic tendencies do not occur at the same *speed*. In particular, we will show that as d increases, the difference between $\mu_{\chi(d,\lambda_{d,1})}$ and $\mu_{\chi(d,\lambda_{d,2})}$ actually *increases*. If we take $\mathbf{x}_{d,1}$ to be closer to the origin than $\mathbf{x}_{d,2}$ ($c_1 < c_2$), this means that $\mathbf{x}_{d,1}$ becomes closer to all other points from the data distribution $\mathbf{Z}_d$ than $\mathbf{x}_{d,2}$, simply by virtue of increasing dimensionality, since for different values of d we place the two points at analogous positions in the data space with regards to the distance from the origin.

5.1.3 ASYMPTOTIC EQUIVALENCE

Before describing our main theoretical result, we present several definitions and lemmas, beginning with the notion of asymptotic equivalence that will be relied upon.

Definition 4 *Two real-valued functions $f(x)$ and $g(x)$ are* asymptotically equal, *$f(x) \approx g(x)$, iff for every $\varepsilon > 0$ there exists $x_0 \in \mathbb{R}$ such that for every $x > x_0$, $|f(x) - g(x)| < \varepsilon$.*

Equivalently, $f(x) \approx g(x)$ iff $\lim_{x\to\infty} |f(x) - g(x)| = 0$. Note that the $\approx$ relation is different from the divisive notion of asymptotic equivalence, where $f(x) \sim g(x)$ iff $\lim_{x\to\infty} f(x)/g(x) = 1$.

The following two lemmas describe approximations that will be used in the proof of Theorem 1, based on the $\approx$ relation.

Lemma 5 *For any constant $c \in \mathbb{R}$, let $f(d) = \sqrt{d+c}$, and $g(d) = \sqrt{d}$. Then, $f(d) \approx g(d)$.*

Proof

$$\lim_{d\to\infty} \left|\sqrt{d+c} - \sqrt{d}\right| = \lim_{d\to\infty} \left|\left(\sqrt{d+c} - \sqrt{d}\right) \frac{\sqrt{d+c}+\sqrt{d}}{\sqrt{d+c}+\sqrt{d}}\right| = \lim_{d\to\infty} \left|\frac{c}{\sqrt{d+c}+\sqrt{d}}\right| = 0. \qquad \blacksquare$$

Lemma 6 *$\mu_{\chi(d)} \approx \sqrt{d}$, and $\sigma_{\chi(d)} \approx 1/\sqrt{2}$.*

Proof Observe the expression for the mean of the $\chi(d)$ distribution,

$$\mu_{\chi(d)} = \sqrt{2}\,\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}.$$

The equality $x\Gamma(x) = \Gamma(x+1)$ and the convexity of $\log\Gamma(x)$ yield (Haagerup, 1982, p. 237):

$$\Gamma\left(\frac{d+1}{2}\right)^2 \leq \Gamma\left(\frac{d}{2}\right)\Gamma\left(\frac{d+2}{2}\right) = \frac{d}{2}\,\Gamma\left(\frac{d}{2}\right)^2,$$

13. More precisely, the lemmas can be applied only for points $\mathbf{x}'_{d,i}$ that have equal values of all components, since after translation data components need to be i.i.d. Because of the symmetry of the Gaussian distribution, the same expectations and variances of distance distributions are obtained, for every d, with any $\mathbf{x}_{d,i}$ that has the same norm as $\mathbf{x}'_{d,i}$, thereby producing identical asymptotic results.

and

$$\Gamma\left(\frac{d+1}{2}\right)^2 = \frac{d-1}{2}\Gamma\left(\frac{d-1}{2}\right)\Gamma\left(\frac{d+1}{2}\right) \geq \frac{d-1}{2}\Gamma\left(\frac{d}{2}\right)^2,$$

from which we have

$$\sqrt{d-1} \leq \sqrt{2}\frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \leq \sqrt{d}.$$

From Lemma 5 it now follows that $\mu_{\chi(d)} \approx \sqrt{d}$.

Regarding the standard deviation of the $\chi(d)$ distribution, from Lemma 3, taking $\mathcal{F}$ to be the standard normal distribution, we obtain

$$\lim_{d\to\infty} \sigma^2_{\chi(d)} = \frac{\sigma^2_{\chi^2(1)}}{4\mu_{\chi^2(1)}} = \frac{1}{2},$$

since the square of a standard normal random variable follows the chi-square distribution with one degree of freedom, $\chi^2(1)$, whose mean and variance are known: $\mu_{\chi^2(1)} = 1$, $\sigma^2_{\chi^2(1)} = 2$. It now directly follows that $\sigma_{\chi(d)} \approx 1/\sqrt{2}$. ■

5.1.4 Expectation of the Noncentral Chi Distribution

The central notion in Theorem 1 is the noncentral Chi distribution. To express the expectation of the noncentral Chi distribution, the following two definitions are needed, introducing the Kummer confluent hypergeometric function ${}_1F_1$, and the generalized Laguerre function.

Definition 7 (Itô, 1993, p. 1799, Appendix A, Table 19.I)
For $a,b,z \in \mathbb{R}$, the Kummer confluent hypergeometric function ${}_1F_1(a;b;z)$ is given by

$${}_1F_1(a;b;z) = \sum_{k=0}^{\infty} \frac{(a)_k}{(b)_k} \cdot \frac{z^k}{\Gamma(k+1)},$$

where $(\cdot)_k$ is the Pochhammer symbol, $(x)_k = \frac{\Gamma(x+k)}{\Gamma(x)}$.

Definition 8 (Itô, 1993, p. 1811, Appendix A, Table 20.VI)
For $\nu,\alpha,z \in \mathbb{R}$, the generalized Laguerre function $L_\nu^{(\alpha)}(z)$ is given by

$$L_\nu^{(\alpha)}(z) = \frac{\Gamma(\nu+\alpha+1)}{\Gamma(\nu+1)} \cdot \frac{{}_1F_1(-\nu;\alpha+1;z)}{\Gamma(\alpha+1)}.$$

The expectation of the noncentral Chi distribution with d degrees of freedom and noncentrality parameter λ, denoted by $\mu_{\chi(d,\lambda)}$, can now be expressed via the generalized Laguerre function (Oberto and Pennecchi, 2006):

$$\mu_{\chi(d,\lambda)} = \sqrt{\frac{\pi}{2}} L_{1/2}^{(d/2-1)}\left(-\frac{\lambda^2}{2}\right). \tag{2}$$

5.1.5 PROPERTIES OF THE GENERALIZED LAGUERRE FUNCTION

The proof of Theorem 1 will rely on several properties of the generalized Laguerre function. In this section we will review two known properties and prove several additional ones as lemmas.

An important property of the generalized Laguerre function is its infinite differentiability in z, with the result of differentiation again being a generalized Laguerre function:

$$\frac{\partial}{\partial z} L_{\nu}^{(\alpha)}(z) = -L_{\nu-1}^{(\alpha+1)}(z)\,. \tag{3}$$

Another useful property is the following recurrence relation:

$$L_{\nu}^{(\alpha)}(z) = L_{\nu-1}^{(\alpha)}(z) + L_{\nu}^{(\alpha-1)}(z)\,. \tag{4}$$

Lemma 9 *For $\alpha > 0$ and $z < 0$:*
(a) $L_{-1/2}^{(\alpha)}(z)$ *is a positive monotonically increasing function in z, while*
(b) $L_{1/2}^{(\alpha)}(z)$ *is a positive monotonically decreasing concave function in z.*

Proof (a) From Definition 8,

$$L_{-1/2}^{(\alpha)}(z) = \frac{\Gamma(\alpha+1/2)}{\Gamma(1/2)} \cdot \frac{{}_1F_1(1/2;\alpha+1;z)}{\Gamma(\alpha+1)}\,.$$

Since $\alpha > 0$ all three terms involving the Gamma function are positive. We transform the remaining term using the equality (Itô, 1993, p. 1799, Appendix A, Table 19.I):

$${}_1F_1(a;b;z) = e^z\,{}_1F_1(b-a;b;-z)\,, \tag{5}$$

which holds arbitrary $a, b, z \in \mathbb{R}$, obtaining

$${}_1F_1(1/2;\alpha+1;z) = e^z\,{}_1F_1(\alpha+1/2;\alpha+1;-z)\,.$$

From Definition 7 it now directly follows that $L_{-1/2}^{(\alpha)}(z)$ is positive for $\alpha > 0$ and $z < 0$.

To show that $L_{-1/2}^{(\alpha)}(z)$ is monotonically increasing in z, from Equation 3 and Definition 8 we have

$$\frac{\partial}{\partial z} L_{-1/2}^{(\alpha)}(z) = -L_{-3/2}^{(\alpha+1)}(z) = -\frac{\Gamma(\alpha+1/2)}{\Gamma(-1/2)} \cdot \frac{{}_1F_1(3/2;\alpha+2;z)}{\Gamma(\alpha+2)}\,.$$

For $\alpha > 0$ and $z < 0$, from Equation 5 it follows that ${}_1F_1(3/2;\alpha+2;z) > 0$. Since $\Gamma(-1/2) < 0$ and all remaining terms are positive, it follows that $-L_{-3/2}^{(\alpha+1)}(z) > 0$. Thus, $L_{-1/2}^{(\alpha)}(z)$ is monotonically increasing in z.

(b) Proofs that $L_{1/2}^{(\alpha)}(z)$ is positive and monotonically decreasing are very similar to the proofs in part (a), and will be omitted. To address concavity, we observe the second derivative of $L_{1/2}^{(\alpha)}(z)$:

$$\frac{\partial^2}{\partial z} L_{1/2}^{(\alpha)}(z) = L_{-3/2}^{(\alpha+2)}(z) = \frac{\Gamma(\alpha+3/2)}{\Gamma(-1/2)} \cdot \frac{{}_1F_1(3/2;\alpha+3;z)}{\Gamma(\alpha+3)}\,.$$

Similarly to part (a), from Equation 5, Definition 7, and basic properties of the gamma function it follows that $L_{-3/2}^{(\alpha+2)}(z) < 0$ for $\alpha > 0$ and $z < 0$. Thus, $L_{1/2}^{(\alpha)}(z)$ is concave in z. ■

Lemma 10 *For* $\alpha > 0$ *and* $z < 0$, $L_{1/2}^{(\alpha+1)}(z) \approx L_{1/2}^{(\alpha)}(z)$.

Proof From the recurrence relation in Equation 4 we obtain

$$L_{1/2}^{(\alpha+1)}(z) = L_{-1/2}^{(\alpha+1)}(z) + L_{1/2}^{(\alpha)}(z).$$

Therefore, to prove the lemma it needs to be shown that for $z < 0$,

$$\lim_{\alpha\to\infty} L_{-1/2}^{(\alpha)}(z) = 0. \tag{6}$$

From Definition 8 and Equation 5 we have

$$L_{-1/2}^{(\alpha)}(z) = \frac{e^{-z}}{\Gamma(1/2)} \cdot \frac{\Gamma(\alpha+1/2)}{\Gamma(\alpha+1)} \cdot {}_1F_1(\alpha+1/2;\alpha+1;-z).$$

From the asymptotic expansion by Fujikoshi (2007, p. 16, adapted),

$${}_1F_1\left(\frac{1}{2}n;\frac{1}{2}(n+b);x\right) = e^x\left(1+O(n^{-1})\right), \tag{7}$$

where n is large and $x \geq 0$, it follows that $\lim_{\alpha\to\infty} {}_1F_1(\alpha+1/2;\alpha+1;-z) < \infty$. Thus, to prove Equation 6 and the lemma it remains to be shown that

$$\lim_{\alpha\to\infty} \frac{\Gamma(\alpha+1/2)}{\Gamma(\alpha+1)} = 0. \tag{8}$$

As in the proof of Lemma 6, from the inequalities derived by Haagerup (1982) we have

$$\sqrt{\beta-1} \leq \sqrt{2}\,\frac{\Gamma\left(\frac{\beta+1}{2}\right)}{\Gamma\left(\frac{\beta}{2}\right)} \leq \sqrt{\beta},$$

where $\beta > 1$. Applying inversion and substituting β with $2\alpha+1$ yields the desired limit. ■

Lemma 11 *For* $\alpha > 1/2$ *and* $z < 0$:
(a) $\lim_{z\to-\infty} L_{-3/2}^{(\alpha)}(z) = 0$, *and*
(b) $\lim_{\alpha\to\infty} L_{-3/2}^{(\alpha)}(z) = 0$.

Proof (a) From Definition 8 we have

$$L_{-3/2}^{(\alpha)}(z) = \frac{\Gamma(\alpha-1/2)}{\Gamma(-1/2)} \cdot \frac{{}_1F_1(3/2;\alpha+1;z)}{\Gamma(\alpha+1)}. \tag{9}$$

The following property (Abramowitz and Stegun, 1964, p. 504, Equation 13.1.5),

$${}_1F_1(a;b;z) = \frac{\Gamma(b)}{\Gamma(b-a)}(-z)^a\left(1+O(|z|^{-1})\right) \quad (z<0),$$

when substituted into Equation 9, taking $a = 3/2$ and $b = \alpha + 1$, yields

$$L_{-3/2}^{(\alpha)}(z) = \frac{1}{\Gamma(-1/2)}(-z)^{-3/2}\left(1 + O(|z|^{-1})\right). \tag{10}$$

From Equation 10 the desired limit directly follows.

(b) The proof of part (b) is analogous to the proof of Lemma 10, that is, Equation 6. From Definition 8 and Equation 5, after applying the expansion by Fujikoshi (2007) given in Equation 7, it remains to be shown that

$$\lim_{\alpha\to\infty} \frac{\Gamma(\alpha - 1/2)}{\Gamma(\alpha + 1)} = 0. \tag{11}$$

Since $\Gamma(\alpha - 1/2) < \Gamma(\alpha + 1/2)$ for every $\alpha \geq 2$, the desired limit in Equation 11 follows directly from Equation 8. ■

5.1.6 THE MAIN RESULT

This section restates and proves our main theoretical result.

Theorem 1 *Let $\lambda_{d,1} = \mu_{\chi(d)} + c_1\sigma_{\chi(d)}$ and $\lambda_{d,2} = \mu_{\chi(d)} + c_2\sigma_{\chi(d)}$, where $d \in \mathbb{N}^+$, $c_1, c_2 \leq 0$, $c_1 < c_2$, and $\mu_{\chi(d)}$ and $\sigma_{\chi(d)}$ are the mean and standard deviation of the Chi distribution with d degrees of freedom, respectively. Define*

$$\Delta\mu_d(\lambda_{d,1}, \lambda_{d,2}) = \mu_{\chi(d,\lambda_{d,2})} - \mu_{\chi(d,\lambda_{d,1})},$$

where $\mu_{\chi(d,\lambda_{d,i})}$ is the mean of the noncentral Chi distribution with d degrees of freedom and noncentrality parameter $\lambda_{d,i}$ ($i \in \{1,2\}$).

There exists $d_0 \in \mathbb{N}$ such that for every $d > d_0$,

$$\Delta\mu_d(\lambda_{d,1}, \lambda_{d,2}) > 0, \tag{12}$$

and

$$\Delta\mu_{d+2}(\lambda_{d+2,1}, \lambda_{d+2,2}) > \Delta\mu_d(\lambda_{d,1}, \lambda_{d,2}). \tag{13}$$

Proof To prove Equation 12, we observe that for $d > 2$,

$$\begin{aligned}
\Delta\mu_d(\lambda_{d,1}, \lambda_{d,2}) &= \mu_{\chi(d,\lambda_{d,2})} - \mu_{\chi(d,\lambda_{d,1})} \\
&= \sqrt{\frac{\pi}{2}}\, L_{1/2}^{(\frac{d}{2}-1)}\left(-\frac{\lambda_{d,2}^2}{2}\right) - \sqrt{\frac{\pi}{2}}\, L_{1/2}^{(\frac{d}{2}-1)}\left(-\frac{\lambda_{d,1}^2}{2}\right) \\
&> 0,
\end{aligned}$$

where the last inequality holds for $d > 2$ because $\lambda_{d,1} < \lambda_{d,2}$, and $L_{1/2}^{(d/2-1)}(z)$ is a monotonically decreasing function in $z < 0$ for $d/2 - 1 > 0$ (Lemma 9).

In order to prove Equation 13, we will use approximate values of noncentrality parameters $\lambda_{d,1}$ and $\lambda_{d,2}$. Let $\widehat{\lambda}_{d,1} = \sqrt{d} + c_1/\sqrt{2}$, and $\widehat{\lambda}_{d,2} = \sqrt{d} + c_2/\sqrt{2}$. From Lemma 6 it follows that $\widehat{\lambda}_{d,1} \approx \lambda_{d,1}$ and $\widehat{\lambda}_{d,2} \approx \lambda_{d,2}$. Thus, by proving that there exists $d_2 \in \mathbb{N}$ such that for every $d > d_2$,

$$\Delta\mu_{d+2}(\widehat{\lambda}_{d+2,1}, \widehat{\lambda}_{d+2,2}) > \Delta\mu_d(\widehat{\lambda}_{d,1}, \widehat{\lambda}_{d,2}), \tag{14}$$

we prove that there exists $d_1 \in \mathbb{N}$ such that for every $d > d_1$ Equation 13 holds. The existence of such d_1, when approximations are used as function arguments, is ensured by the fact that $L_{1/2}^{(\alpha)}(z)$ is a monotonically decreasing *concave* function in z (Lemma 9), and by the transition from α to $\alpha+1$ having an insignificant impact on the value of the Laguerre function for large enough α (Lemma 10). Once Equation 14 is proven, Equations 12 and 13 will hold for every $d > d_0$, where $d_0 = \max(2, d_1)$.

To prove Equation 14, from Equation 2 it follows we need to show that

$$\begin{aligned} & L_{1/2}^{(d/2)}\left(-\frac{1}{2}\left(\sqrt{d+2}+c_2/\sqrt{2}\right)^2\right) - L_{1/2}^{(d/2)}\left(-\frac{1}{2}\left(\sqrt{d+2}+c_1/\sqrt{2}\right)^2\right) \\ & \quad > \; L_{1/2}^{(d/2-1)}\left(-\frac{1}{2}\left(\sqrt{d}+c_2/\sqrt{2}\right)^2\right) - L_{1/2}^{(d/2-1)}\left(-\frac{1}{2}\left(\sqrt{d}+c_1/\sqrt{2}\right)^2\right). \end{aligned} \tag{15}$$

We observe the second derivative of $L_{1/2}^{(\alpha)}(z)$:

$$\frac{\partial^2}{\partial z} L_{1/2}^{(\alpha)}(z) = L_{-3/2}^{(\alpha+2)}(z).$$

Since $L_{-3/2}^{(\alpha+2)}(z)$ tends to 0 as $z \to -\infty$, and tends to 0 also as $\alpha \to \infty$ (Lemma 11), it follows that the two Laguerre functions on the left side of Equation 15 can be approximated by a linear function with an arbitrary degree of accuracy for large enough d. More precisely, since $L_{1/2}^{(\alpha)}(z)$ is monotonically decreasing in z (Lemma 9) there exist $a, b \in \mathbb{R}$, $a > 0$, such that the left side of Equation 15, for large enough d, can be replaced by

$$\begin{aligned} & -a\left(-\frac{1}{2}\left(\sqrt{d+2}+c_2/\sqrt{2}\right)^2\right)+b - \left(-a\left(-\frac{1}{2}\left(\sqrt{d+2}+c_1/\sqrt{2}\right)^2\right)+b\right) \\ & \quad = \; \frac{a}{2}\left(\sqrt{d+2}+c_2/\sqrt{2}\right)^2 - \frac{a}{2}\left(\sqrt{d+2}+c_1/\sqrt{2}\right)^2. \end{aligned} \tag{16}$$

From Lemma 10 it follows that the same linear approximation can be used for the right side of Equation 15, replacing it by

$$\frac{a}{2}\left(\sqrt{d}+c_2/\sqrt{2}\right)^2 - \frac{a}{2}\left(\sqrt{d}+c_1/\sqrt{2}\right)^2. \tag{17}$$

After substituting the left and right side of Equation 15 with Equations 16 and 17, respectively, it remains to be shown that

$$\begin{aligned} & \frac{a}{2}\left(\sqrt{d+2}+c_2/\sqrt{2}\right)^2 - \frac{a}{2}\left(\sqrt{d+2}+c_1/\sqrt{2}\right)^2 \\ & \quad > \frac{a}{2}\left(\sqrt{d}+c_2/\sqrt{2}\right)^2 - \frac{a}{2}\left(\sqrt{d}+c_1/\sqrt{2}\right)^2. \end{aligned} \tag{18}$$

Multiplying both sides by $\sqrt{2}/a$, moving the right side to the left, and applying algebraic simplification reduces Equation 18 to

$$\left(c_2 - c_1\right)\left(\sqrt{d+2} - \sqrt{d}\right) > 0,$$

which holds for $c_1 < c_2$, thus concluding the proof. ■

5.2 Discussion

This section will discuss several additional considerations and related work regarding the geometry of high-dimensional spaces and the behavior of data distributions within them. First, let us consider the geometric upper limit to the number of points that point $\mathbf{x}$ can be a nearest neighbor of, in Euclidean space. In one dimension, this number is 2, in two dimensions it is 5, while in 3 dimensions it equals 11 (Tversky and Hutchinson, 1986). Generally, for Euclidean space of dimensionality d this number is equal to the *kissing number*, which is the maximal number of hyperspheres that can be placed to touch a given hypersphere without overlapping, with all hyperspheres being of the same size.[14] Exact kissing numbers for arbitrary d are generally not known, however there exist bounds which imply that they progress exponentially with d (Odlyzko and Sloane, 1979; Zeger and Gersho, 1994). Furthermore, when considering k nearest neighbors for $k > 1$, the bounds become even larger. Therefore, only for very low values of d geometrical constraints of vector space prevent hubness. On the other hand, for higher values of d hubness may or may not occur, and the geometric bounds, besides providing "room" for hubness (even for values of k as low as 1) do not contribute much in fully characterizing the hubness phenomenon. Therefore, in high dimensions the behavior of *data distributions* needed to be studied.

We focus the rest of the discussion around the following important result,[15] drawing parallels with our results and analysis, and extending existing interpretations.

Theorem 12 (Newman and Rinott, 1985, p. 803, Theorem 3, adapted)

Let $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_d^{(i)})$, $i = 0, \ldots, n$ *be a sample of* $n+1$ *i.i.d. points from distribution* $\mathbf{F}(\mathbf{X})$, $\mathbf{X} = (X_1, \ldots, X_d) \in \mathbb{R}^d$. *Assume that* $\mathbf{F}$ *is of the form* $\mathbf{F}(\mathbf{X}) = \prod_{k=1}^{d} \mathcal{F}(X_k)$, *that is, the coordinates* $X_1, \ldots, X_d$ *are i.i.d. Let the distance measure be of the form* $D(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sum_{k=1}^{d} g(x_k^{(i)}, x_k^{(j)})$. *Let* $N_1^{n,d}$ *denote the number of points among* $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ *whose nearest neighbor is* $\mathbf{x}^{(0)}$.

Suppose $0 < \mathrm{Var}(g(X,Y)) < \infty$ *and set*

$$\beta = \mathrm{Correlation}(g(X,Y), g(X,Z)), \tag{19}$$

where X, Y, Z *are i.i.d. with common distribution* $\mathcal{F}$ *(the marginal distribution of* X_k*).*
(a) *If* $\beta = 0$ *then*

$$\lim_{n\to\infty} \lim_{d\to\infty} N_1^{n,d} = \mathrm{Poisson}(\lambda = 1) \ \textit{in distribution} \tag{20}$$

and

$$\lim_{n\to\infty} \lim_{d\to\infty} \mathrm{Var}(N_1^{n,d}) = 1. \tag{21}$$

(b) *If* $\beta > 0$ *then*

$$\lim_{n\to\infty} \lim_{d\to\infty} N_1^{n,d} = 0 \ \textit{in distribution} \tag{22}$$

while

$$\lim_{n\to\infty} \lim_{d\to\infty} \mathrm{Var}(N_1^{n,d}) = \infty. \tag{23}$$

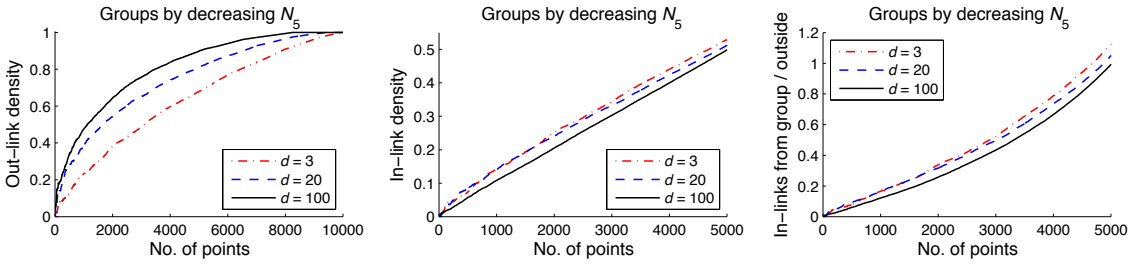
14. If ties are disallowed, it may be necessary to subtract 1 from the kissing number to obtain the maximum of N_1.
15. A theorem that is effectively a special case of this result was proven previously (Newman et al., 1983, Theorem 7) for continuous distributions with finite kurtosis and Euclidean distance.

What is exceptional in this theorem are Equations 22 and 23. According to the interpretation by Tversky et al. (1983), they suggest that if the number of dimensions is large relative to the number of points, one may expect to have a large proportion of points with N_1 equaling 0, and a small proportion of points with high N_1 values, that is, hubs.[16] Trivially, Equation 23 also holds for N_k with $k > 1$, since for any point $\mathbf{x}$, $N_k(\mathbf{x}) \geq N_1(\mathbf{x})$.

The setting involving i.i.d. normal random data and Euclidean distance, used in our Theorem 1 (and, generally, any i.i.d. random data distribution with Euclidean distance), fulfills the conditions of Theorem 12 for Equations 22 and 23 to be applied, since the correlation parameter $\beta > 0$. This correlation exists because, for example, if we view vector component variable X_j ($j \in \{1, 2, \ldots, d\}$) and the distribution of data points within it, if a random point drawn from X_j is closer to the mean of X_j it is more likely to be close to other random points drawn from X_j, and vice versa, producing the case $\beta > 0$.[17] Therefore, Equations 22 and 23 from Theorem 12 directly apply to the setting studied in Theorem 1, providing asymptotic evidence for hubness. However, since the proof of Equations 22 and 23 in Theorem 12 relies on applying the central limit theorem to the (normalized) distributions of pairwise distances between vectors $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ ($0 \leq i \neq j \leq n$) as $d \to \infty$ (obtaining limit distance distributions which are Gaussian), the results of Theorem 12 are inherently asymptotic in nature. Theorem 1, on the other hand, describes the behavior of distances in finite dimensionalities,[18] providing the means to characterize the behavior of N_k in high, but finite-dimensional space. What remains to be done is to formally connect Theorem 1 with the skewness of N_k in finite dimensionalities, for example by observing point $\mathbf{x}$ with a fixed position relative to the data distribution mean (the origin) across dimensionalities, in terms of being at distance $\mu_{\chi(d)} + c\,\sigma_{\chi(d)}$ from the origin, and expressing how the probability of $\mathbf{x}$ to be the nearest neighbor (or among the k nearest neighbors) of a randomly drawn point changes with increasing dimensionality.[19] We address this investigation as a point of future work.

Returning to Theorem 12 and the value of β from Equation 19, as previously discussed, $\beta > 0$ signifies that the position of a vector component value makes a difference when computing distances between vectors, causing some component values to be more "special" than others. Another contribution of Theorem 1 is that it illustrates how the individual differences in component values combine to make positions of whole vectors more special (by being closer to the data center). On the other hand, if $\beta = 0$ no point can have a special position with respect to all others. In this case, Equations 20 and 21 hold, which imply there is no hubness. This setting is relevant, for example, to points being generated by a Poisson process which spreads the vectors uniformly over $\mathbb{R}^d$, where all positions within the space (both at component-level and globally) become basically equivalent. Although it does not directly fit into the framework of Theorem 12, the same principle can be used to explain the absence of hubness for normally distributed data and cosine distance from Section 3.1: in this setting no vector is more spatially central than any other. Equations 20 and 21, which imply no hubness, hold for many more "centerless" settings, including random graphs, settings with exchangeable distances, and d-dimensional toruses (Newman and Rinott, 1985).

16. Reversing the order of limits, which corresponds to having a large number of points relative to the number of dimensions, produces the same asymptotic behavior as in Equations 20 and 21, that is, no hubness, in all studied settings.
17. Similarly to Section 4.2, this argument holds for unimodal distributions of component variables; for multimodal distributions the driving force behind nonzero β is the proximity to a peak in the probability density function.
18. Although the statement of Theorem 1 relies on dimensionality being greater than some d_0 which is finite, but can be arbitrarily high, empirical evidence suggests that actual d_0 values are low, often equaling 0.
19. For $c < 0$ we expect this probability to increase since $\mathbf{x}$ is closer to the data distribution mean, and becomes closer to other points as dimensionality increases.



(a) (b) (c)

Figure 8: (a) Out-link, and (b) in-link densities of groups of hubs with increasing size; (c) ratio of the number of in-links originating from points within the group and in-links originating from outside points, for i.i.d. uniform random data with dimensionality $d = 3,\ 20,\ 100$.

The following two subsections will address several additional issues concerning the interpretation of Theorem 12.

5.2.1 NEAREST-NEIGHBOR GRAPH STRUCTURE

The interpretation of Theorem 12 by Tversky et al. (1983) may be understood in the sense that, with increasing dimensionality, *very few* exceptional points become hubs, while all others are relegated to antihubs. In this section we will empirically examine the structural change of the k-NN graph as the number of dimensions increases. We will also discuss and consolidate different notions node centrality in the k-NN graph, and their dependence on the (intrinsic) dimensionality of data.

First, as in Section 3.1, we consider $n = 10000$ i.i.d. uniform random data points of different dimensionality. Let us observe hubs, that is, points with highest N_5, collected in groups of progressively increasing size: $5, 10, 15, \ldots, 10000$. In analogy with the notion of network density from social network analysis (Scott, 2000), we define group *out-link density* as the proportion of the number of arcs that originate and end in nodes from the group, and the total number of arcs that originate from nodes in the group. Conversely, we define group *in-link density* as the proportion of the number of arcs that originate and end in nodes from the group, and the total number of arcs that *end* in nodes from the group. Figure 8(a, b) shows the out-link and in-link densities of groups of strongest hubs in i.i.d. uniform random data (similar tendencies can be observed with other synthetic data distributions). It can be seen in Figure 8(a) that hubs are more cohesive in high dimensions, with more of their out-links leading to other hubs. On the other hand, Figure 8(b) suggests that hubs also receive more in-links from non-hub points in high dimensions than in low dimensions. Moreover, Figure 8(c), which plots the ratio of the number of in-links that originate within the group, and the number of in-links which originate outside, shows that hubs receive a larger proportion of in-links from non-hub points in high dimensions than in low dimensions. We have reported our findings for $k = 5$, however similar results are obtained with other values of k.

Overall, it can be said that in high dimensions hubs receive more in-links than in low dimensions from both hubs and non-hubs, and that the range of influence of hubs gradually widens as dimensionality increases. We can therefore conclude that the transition of hubness from low to high dimensionalities is "smooth," both in the sense of the change in the overall distribution of N_k, and the change in the degree of influence of data points, as expressed by the above analysis of links.

So far, we have viewed hubs primarily through their exhibited high values of N_k, that is, high degree centrality in the k-NN directed graph. However, (scale-free) network analysis literature often attributes other properties to hubs (Albert and Barabási, 2002), viewing them as nodes that are important for preserving network structure due to their central positions within the *graph*, indicated, for example, by their *betweenness centrality* (Scott, 2000). On the other hand, as discussed by Li et al. (2005), in both synthetic and real-world networks high-degree nodes do not necessarily need to correspond to nodes that are central in the graph, that is, high-degree nodes can be concentrated at the *periphery* of the network and bear little structural significance. For this reason, we computed the betweenness centrality of nodes in k-NN graphs of synthetic and real data sets studied in this paper, and calculated its Spearman correlation with node degree, denoting the measure by $C_{\mathrm{BC}}^{N_k}$. For i.i.d. uniform data ($k=5$), when $d=3$ the measured correlation is $C_{\mathrm{BC}}^{N_5}=0.311$, when $d=20$ the correlation is $C_{\mathrm{BC}}^{N_5}=0.539$, and finally when $d=100$ the correlation rises to $C_{\mathrm{BC}}^{N_5}=0.647$.[20] This suggests that with increasing dimensionality the centrality of nodes increases not only in the sense of higher node degree or spatial centrality of vectors (as discussed in Section 4.2), but also in the structural graph-based sense. We support this observation further by computing, over the 50 real data sets listed in Table 1, the correlation between $C_{\mathrm{BC}}^{N_{10}}$ and $S_{N_{10}}$, finding it to be significant: 0.548.[21] This indicates that real data sets which exhibit strong skewness in the distribution of N_{10} also tend to have strong correlation between N_{10} and betweenness centrality of nodes, giving hubs a broader significance for the structure of the k-NN graphs.

5.2.2 Rate of Convergence and the Role of Boundaries

On several occasions, the authors of Theorem 12 have somewhat downplayed the significance of equations 22 and 23 (Tversky et al., 1983; Newman and Rinott, 1985), citing empirically observed slow convergence (Maloney, 1983), even to the extent of not observing significant differences between hubness in the Poisson process and i.i.d. uniform cube settings. However, results in the preceding sections of this paper suggest that this convergence is fast enough to produce notable hubness in high-dimensional data. In order to directly illustrate the difference between a setting which provides no possibility for spatial centrality of points, and one that does, we will observe the Poisson process vs. the i.i.d. unit cube setting. We will be focusing on the location of the nearest neighbor of a point from the cube, that is, on determining whether it stays within the boundaries of the cube as dimensionality increases.

Lemma 13 *Let points be spread in $\mathbb{R}^d$ according to a Poisson process with constant intensity $\lambda > 1$. Observe a unit hypercube $C \subset \mathbb{R}^d$, and an arbitrary point $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in C$, generated by the Poisson process. Let $p_{\lambda,d}$ denote the probability that the nearest neighbor of $\mathbf{x}$, with respect to Euclidean distance, is not situated in C. Then,*

$$\lim_{d\to\infty} p_{\lambda,d} = 1\,.$$

Proof Out of the 3^d-1 unit hypercubes that surround C, let us observe only the $2d$ hypercubes that differ from C only in one coordinate. We will restrict the set of considered points to these $2d$ cubes, and prove that the probability that the nearest neighbor of $\mathbf{x}$ comes from one of the $2d$ cubes, $\widehat{p}_{\lambda,d}$, converges to 1 as $d\to\infty$. From this, the convergence of $p_{\lambda,d}$ directly follows, since $p_{\lambda,d} \geq \widehat{p}_{\lambda,d}$.

20. Betweenness centrality is computed on directed k-NN graphs. We obtained similar correlations when undirected graphs were used.

21. When betweenness centrality is computed on undirected graphs, the correlation is even stronger: 0.585.

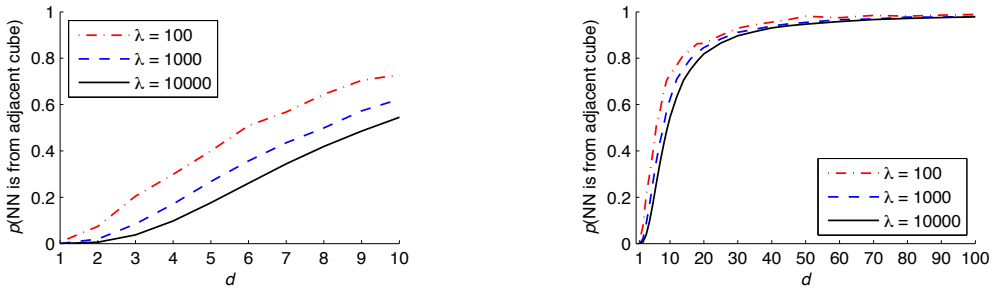


Figure 9: Probability that the nearest neighbor of a point from the unit hypercube originates from one of the adjacent hypercubes, for Poisson processes with $\lambda = 100$, 1000, and 10000 expected points per hypercube, obtained through simulation and averaged over 10 runs.

Let $\widehat{p}_{\lambda,d}(i)$ denote the probability that the nearest neighbor of $\mathbf{x}$ comes from one of the two hypercubes which differ from C only in the ith coordinate, $i \in \{1,2,\ldots,d\}$. For a given coordinate i and point $\mathbf{x}$, let the *1-dimensional nearest neighbor* of x_i denote the closest y_i value of all other points $\mathbf{y}$ from the Poisson process, observed when all coordinates except i are disregarded. Conversely, for a given coordinate i and point $\mathbf{x}$, let the $(d-1)$*-dimensional nearest neighbor* of $(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_d)$ denote the closest point $(y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_d)$, obtained when coordinate i is disregarded.

Observing the ith coordinate only, the probability for the 1-dimensional nearest neighbor of x_i to come from one of the surrounding unit intervals is $\widehat{p}_{\lambda,1}$. Although small, $\widehat{p}_{\lambda,1} > 0$. Assuming this event has occurred, let $\mathbf{y} \in \mathbb{R}^d$ be the point whose component y_i is the 1-dimensional nearest neighbor of x_i that is not within the unit interval containing x_i. Let r_λ denote the probability that the remaining coordinates of $\mathbf{y}$, $(y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_d)$, constitute a $(d-1)$-dimensional nearest neighbor of $(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_d)$, within the confines of C. It can be observed that r_λ is strictly greater than 0, inversely proportional to λ (roughly equaling $1/(\lambda-1)$), and independent of d. Thus, $\widehat{p}_{\lambda,d}(i) \geq \widehat{p}_{\lambda,1} \cdot r_\lambda > 0$.

Let $\widehat{q}_{\lambda,d} = 1 - \widehat{p}_{\lambda,d}$, the probability that the nearest neighbor of $\mathbf{x}$ comes from C (recall the restriction of the location of the nearest neighbor to C and its immediately surrounding $2d$ hypercubes). In light of the above, $\widehat{q}_{\lambda,d} = \prod_{i=1}^{d} \widehat{q}_{\lambda,d}(i) = \prod_{i=1}^{d} (1 - \widehat{p}_{\lambda,d}(i))$. Since each $\widehat{p}_{\lambda,d}(i)$ is bounded from below by a constant strictly greater than 0 (which depends only on λ), each $\widehat{q}_{\lambda,d}(i)$ is bounded from above by a constant strictly smaller than 1. It follows that $\lim_{d\to\infty} \widehat{q}_{\lambda,d} = 0$, and therefore $\lim_{d\to\infty} \widehat{p}_{\lambda,d} = 1$. ■

To illustrate the rate of convergence in Lemma 13, Figure 9 plots the empirically observed probabilities that the nearest neighbor of a point from a unit hypercube originates in one of the $2d$ immediately adjacent unit hypercubes ($\widehat{p}_{\lambda,d}$). It can be seen that the probability that the nearest neighbor comes from outside of the cube quickly becomes close to 1 as dimensionality increases. Please note that due to feasibility of simulation the plots in Figure 9 represent the empirical lower bounds (that is, the empirical estimates of $\widehat{p}_{\lambda,d}$) of the true probabilities from Lemma 13 ($p_{\lambda,d}$) by considering only the $2d$ immediately adjacent hypercubes.

The above indicates that when boundaries are introduced in high dimensions, the setting completely changes, in the sense that new nearest neighbors need to be located inside the boundaries. Under such circumstances, points which are closer to the center have a better chance of becoming nearest neighbors, with the mechanism described in previous sections. Another implication of the above observations, stemming from Figure 9, is that the number of dimensions does not need to be very large compared to the number of points for the setting to change. As for boundaries, they can be viewed as a dual notion to spatial centrality discussed earlier. With Poisson processes and cubes this duality is rather straightforward, however for continuous distributions in general there exist no boundaries in a strict mathematical sense. Nevertheless, since data sets contain a finite number of points, it can be said that "practical" boundaries exist in this case as well.

6. Hubness and Dimensionality Reduction

In this section we elaborate further on the interplay of skewness of N_k and intrinsic dimensionality by considering dimensionality-reduction (DR) techniques. The main question motivating the discussion in this section is whether dimensionality reduction can alleviate the issue of the skewness of k-occurrences altogether. We leave a more detailed and general investigation of the interaction between hubness and dimensionality reduction as a point of future work.

We examined the following methods: principal component analysis—PCA (Jolliffe, 2002), independent component analysis—ICA (Hyvärinen and Oja, 2000), stochastic neighbor embedding—SNE (Hinton and Roweis, 2003), isomap (Tenenbaum et al., 2000), and diffusion maps (Lafon and Lee, 2006; Nadler et al., 2006). Figure 10 depicts the relationship between the percentage of the original number of features maintained by the DR methods and S_{N_k}, for several high-dimensional real data sets (musk1, mfeat-factors, and spectrometer; see Table 1) and i.i.d. uniform random data (with Euclidean distance, $k = 10$, and the same number of neighbors used for isomap and diffusion maps). For PCA, ICA, SNE, and the real data sets, looking from right to left, S_{N_k} stays relatively constant until a small percentage of features is left, after which it suddenly drops (Figure 10(a–c)). It can be said that this is the point where the intrinsic dimensionality of data sets is reached, and further reduction of dimensionality may incur loss of valuable information. Such behavior is in contrast with the case of i.i.d. uniform random data (full black line in Figure 10(a–c)), where S_{N_k} steadily and steeply reduces with the decreasing number of randomly selected features (dimensionality reduction is not meaningful in this case), because intrinsic and embedded dimensionalities are equal. Since PCA is equivalent to metric multidimensional scaling (MDS) when Euclidean distances are used (Tenenbaum et al., 2000), and SNE is a variant of MDS which favors the preservation of distances between nearby points, we can roughly regard the notion of intrinsic dimensionality used in this paper as the minimal number of features needed to account for all *pairwise distances* within a data set. Although ICA does not attempt to explicitly preserve pairwise distances, combinations of independent components produce skewness of N_k which behaves in a way that is similar to the skewness observed with PCA and SNE.

On the other hand, isomap and diffusion maps replace the original distances with distances derived from a neighborhood graph. It can be observed in Figure 10(d,e) that such replacement generally reduces S_{N_k}, but in most cases does not alleviate it completely. With the decreasing number of features, however, S_{N_k} of real data sets in Figure 10(d,e) still behaves in a manner more similar to S_{N_k} of real data sets for PCA, ICA, and SNE (Figure 10(a–c)) than i.i.d. random data (dash-dot black line in Figure 10(d,e)).

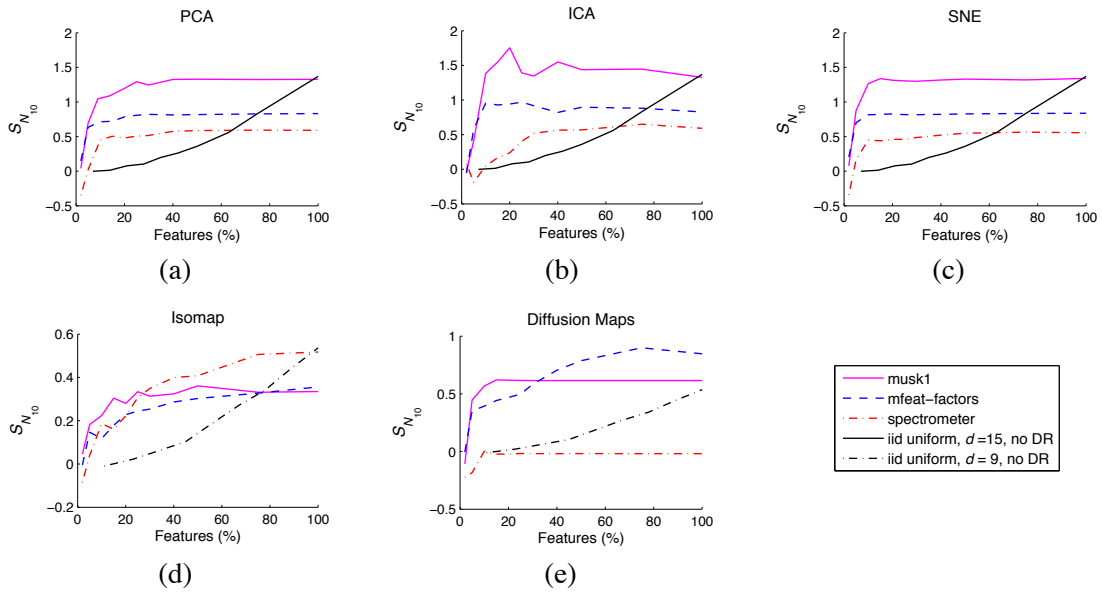


Figure 10: Skewness of N_{10} in relation to the percentage of the original number of features maintained by dimensionality reduction, for real and i.i.d. uniform random data and (a) principal component analysis—PCA, (b) independent component analysis—ICA, (c) stochastic neighbor embedding—SNE, (d) isomap, and (e) diffusion maps.

The above observations signify that, if distances are not explicitly altered (as with isomap and diffusion maps DR methods), that is, if one cares about preserving the original distances, dimensionality reduction may not have a significant effect on hubness when the number of features is above the intrinsic dimensionality. This observation is useful in most practical cases because if dimensionality is reduced below intrinsic dimensionality, loss of information can occur. If one still chooses to apply aggressive dimensionality reduction and let the resulting number of features fall below intrinsic dimensionality, it can be expected of pairwise distances and nearest-neighbor relations between points in the data set to be altered, and hubness to be reduced or even lost. Whether these effects should be actively avoided or sought really depends on the application domain and task at hand, that is, whether and to what degree the original pairwise distances represent valuable information, and how useful are the new distances and neighborhoods after dimensionality reduction.

7. The Impact of Hubness on Machine Learning

The impact of hubness on machine-learning applications has not been thoroughly investigated so far. In this section we examine a wide range of commonly used machine-learning methods for supervised (Section 7.1), semi-supervised (Section 7.2), and unsupervised learning (Section 7.3), that either directly or indirectly use distances in the process of building a model. Our main objective is to demonstrate that hubs (as well as their opposites, antihubs) can have a significant effect on these methods. The presented results highlight the need to take hubs into account in a way equivalent to other factors, such as the existence of outliers, the role of which has been well studied.

7.1 Supervised Learning

To investigate possible implications of hubness on supervised learning, we first study the interaction of k-occurrences with information provided by labels (Section 7.1.1). We then move on to examine the effects of hubness on several well-known classification algorithms in Section 7.1.2.

7.1.1 "GOOD" AND "BAD" k-OCCURRENCES

When labels are present, k-occurrences can be distinguished based on whether labels of neighbors match. We define the number of *"bad" k-occurrences* of $\mathbf{x}$, $BN_k(\mathbf{x})$, as the number of points from data set D for which $\mathbf{x}$ is among the first k NNs, and the labels of $\mathbf{x}$ and the points in question *do not match*. Conversely, $GN_k(\mathbf{x})$, the number of *"good" k-occurrences* of $\mathbf{x}$, is the number of such points where labels do match. Naturally, for every $\mathbf{x} \in D$, $N_k(\mathbf{x}) = BN_k(\mathbf{x}) + GN_k(\mathbf{x})$.

To account for labels, Table 1 includes $\widetilde{BN}_{10}$ (14th column), the sum of all observed "bad" k-occurrences of a data set normalized by $\sum_{\mathbf{x}} N_{10}(\mathbf{x}) = 10n$. This measure is intended to express the total amount of "bad" k-occurrences within a data set. Also, to express the amount of information that "regular" k-occurrences contain about "bad" k-occurrences in a particular data set, $C_{BN_{10}}^{N_{10}}$ (15th column) denotes the Spearman correlation between BN_{10} and N_{10} vectors. The motivation behind this measure is to express the degree to which BN_k and N_k follow a similar distribution.

"Bad" hubs, that is, points with high BN_k, are of particular interest to supervised learning because they carry more information about the location of the decision boundaries than other points, and affect classification algorithms in different ways (as will be described in Section 7.1.2). To understand the origins of "bad" hubs in real data, we rely on the notion of the *cluster assumption* from semi-supervised learning (Chapelle et al., 2006), which roughly states that most pairs of points in a high density region (cluster) should be of the same class. To measure the degree to which the cluster assumption is violated in a particular data set, we simply define the *cluster assumption violation* (CAV) coefficient as follows. Let a be the number of pairs of points which are in different classes but in the same cluster, and b the number of pairs of points which are in the same class and cluster. Then, we define

$$\mathrm{CAV} = \frac{a}{a+b},$$

which gives a number in the $[0,1]$ range, higher if there is more violation. To reduce the sensitivity of CAV to the number of clusters (too low and it will be overly pessimistic, too high and it will be overly optimistic), we choose the number of clusters to be 3 times the number of classes of a particular data set. Clustering is performed with K-means.

For all examined real data sets, we computed the Spearman correlation between the total amount of "bad" k-occurrences, $\widetilde{BN}_{10}$, and CAV (16th column of Table 1) and found it strong (0.85, see Table 2). Another significant correlation (0.39) is observed between $C_{BN_{10}}^{N_{10}}$ and intrinsic dimensionality. In contrast, $\widetilde{BN}_{10}$ and CAV are not correlated with intrinsic dimensionality nor with the skewness of N_{10}. The latter fact indicates that high dimensionality and skewness of N_k are not sufficient to induce "bad" hubs. Instead, based on the former fact, we can argue that there are two, mostly independent, forces at work: violation of the cluster assumption on one hand, and high intrinsic dimensionality on the other. "Bad" hubs originate from putting the two together; that is, the consequences of violating the cluster assumption can be more severe in high dimensions than in low dimensions, not in terms of the total amount of "bad" k-occurrences, but in terms of their distribution, since strong regular hubs are now more prone to "pick up" bad k-occurrences than non-hub points. This is supported by

the positive correlation between $C_{BN_{10}}^{N_{10}}$ and intrinsic dimensionality, meaning that in high dimensions BN_k tends to follow a more similar distribution to N_k than in low dimensions.

7.1.2 INFLUENCE ON CLASSIFICATION ALGORITHMS

We now examine how skewness of N_k and the existence of ("bad") hubs affects well-known classification techniques, focusing on the k-nearest neighbor classifier (k-NN), support vector machines (SVM), and AdaBoost. We demonstrate our findings on a selection of data sets from Table 1 which have relatively high (intrinsic) dimensionality, and a non-negligible amount of "badness" ($\widetilde{BN}_k$) and cluster assumption violation (CAV). Generally, the examined classification algorithms (including semi-supervised learning from Section 7.2) exhibit similar behavior on other data sets from Table 1 with the aforementioned properties, and also with various different values of k (in the general range 1–50, as we focused on values of k which are significantly smaller than the number of points in a data set).

k-nearest neighbor classifier. The k-nearest neighbor classifier is negatively affected by the presence of "bad" hubs, because they provide erroneous class information to many other points. To validate this assumption, we devised a simple weighting scheme. For each point $\mathbf{x}$, we compute its standardized "bad" hubness score:

$$h_B(\mathbf{x},k) = \frac{BN_k(\mathbf{x}) - \mu_{BN_k}}{\sigma_{BN_k}},$$

where μ_{BN_k} and σ_{BN_k} are the mean and standard deviation of BN_k, respectively. During majority voting in the k-NN classification phase, when point $\mathbf{x}$ participates in the k-NN list of the point being classified, the vote of $\mathbf{x}$ is weighted by

$$w_k(\mathbf{x}) = \exp(-h_B(\mathbf{x},k)),$$

thus lowering the influence of "bad" hubs on the classification decision. Figure 11 compares the resulting accuracy of k-NN classifier with and without this weighting scheme for six data sets from Table 1. Leave-one-out cross-validation is performed, with Euclidean distance being used for determining nearest neighbors. The k value for N_k is naturally set to the k value used by the k-NN classifier, and $h_B(\mathbf{x},k)$ is recomputed for the training set of each fold. The reduced accuracy of the unweighted scheme signifies the negative influence of "bad" hubs.

Although "bad" hubs tend to carry more information about the location of class boundaries than other points, the "model" created by the k-NN classifier places the emphasis on describing non-borderline regions of the space occupied by each class. For this reason, it can be said that "bad" hubs are truly bad for k-NN classification, creating the need to penalize their influence on the classification decision. On the other hand, for classifiers that explicitly model the borders between classes, such as support vector machines, "bad" hubs can represent points which contribute information to the model in a positive way, as will be discussed next.

Support vector machines. We consider SVMs with the RBF (Gaussian) kernel of the form:

$$K(\mathbf{x},\mathbf{y}) = \exp\left(-\gamma\|\mathbf{x}-\mathbf{y}\|^2\right),$$

where γ is a data-dependent constant. $K(\mathbf{x},\mathbf{y})$ is a smooth monotone function of Euclidean distance between points. Therefore, N_k values in the kernel space are exactly the same as in the original

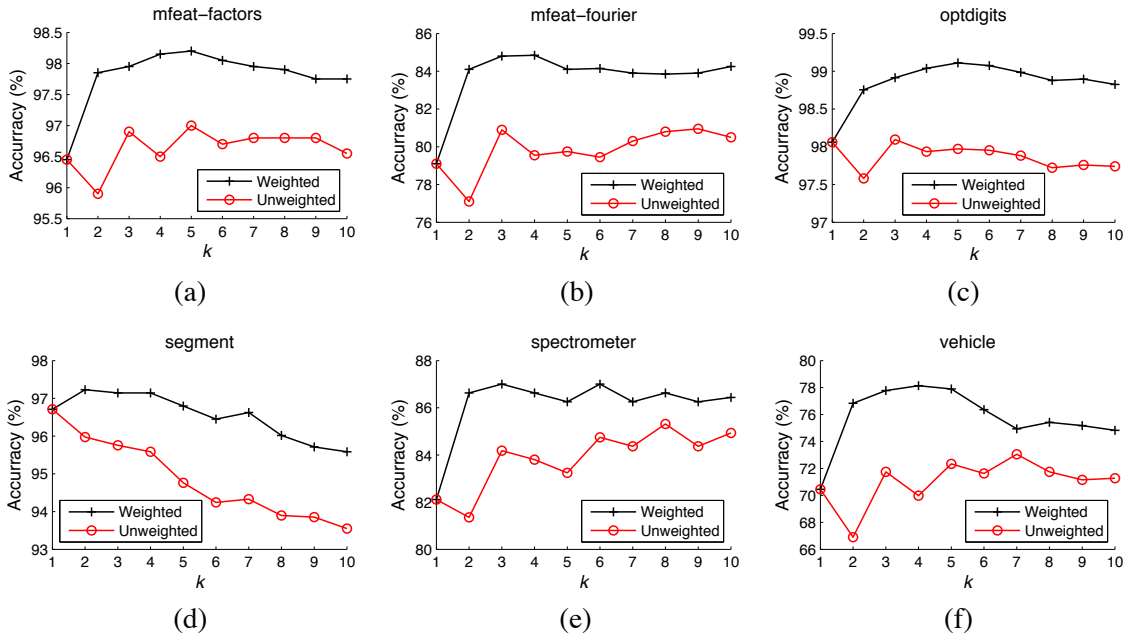


Figure 11: Accuracy of k-NN classifier with and without the weighting scheme.

space.[22] To examine the influence of "bad" hubs on SVMs, Figure 12 illustrates 10-fold cross-validation accuracy results of SVM trained using sequential minimal optimization (Platt, 1999; Keerthi et al., 2001), when points are progressively removed from the training sets: (i) by decreasing BN_k ($k = 5$), and (ii) at random. Accuracy drops with removal by BN_k, indicating that bad hubs are important for support vector machines. The difference in SVM accuracy between random removal and removal by BN_k becomes consistently significant at some stage of progressive removal, as denoted by the dashed vertical lines in the plots, according to the paired t-test at 0.05 significance level.[23]

The reason behind the above observation is that for high-dimensional data, points with high BN_k can comprise good support vectors. Table 3 exemplifies this point by listing the normalized average ranks of support vectors in the 10-fold cross-validation models with regards to decreasing BN_k. The ranks are in the range $[0,1]$, with the value 0.5 expected from a random set of points. Lower values of the ranks indicate that the support vectors, on average, tend to have high BN_k. The table also lists the values of the γ parameter of the RBF kernel, as determined by independent experiments involving 9-fold cross-validation.

AdaBoost. Boosting algorithms take into account the "importance" of points in the training set for classification by weak learners, usually by assigning and updating weights of individual points—the higher the weight, the more attention is to be paid to the point by subsequently trained weak learners. We consider the AdaBoost.MH algorithm (Schapire and Singer, 1999) in conjunction with

22. Centering the kernel matrix changes the N_k of points in the kernel space, but we observed that the overall distribution (that is, its skewness) does not become radically different. Therefore, the following arguments still hold for centered kernels, providing N_k is computed in the kernel space.
23. Since random removal was performed in 20 runs, fold-wise accuracies for statistical testing were obtained in this case by averaging over the runs.

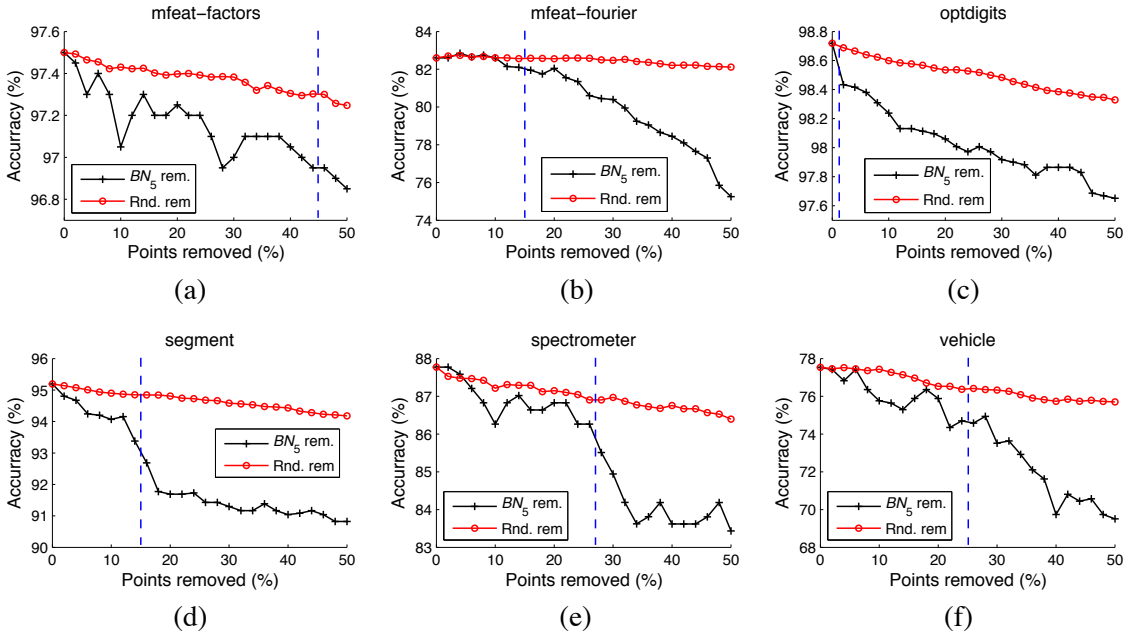


Figure 12: Accuracy of SVM with RBF kernel and points being removed from the training sets by decreasing BN_5, and at random (averaged over 20 runs).

Data set	γ	SV rank	Data set	γ	SV rank
mfeat-factors	0.005	0.218	segment	0.3	0.272
mfeat-fourier	0.02	0.381	spectrometer	0.005	0.383
optdigits	0.02	0.189	vehicle	0.07	0.464

Table 3: Normalized average support vector ranks with regards to decreasing BN_5.

CART trees (Breiman et al., 1984) with the maximal depth of three.[24] We define for each point $\mathbf{x}$ its *standardized hubness* score:

$$h(\mathbf{x},k) = \frac{N_k(\mathbf{x}) - \mu_{N_k}}{\sigma_{N_k}}, \tag{24}$$

where μ_{N_k}, σ_{N_k} are the mean and standard deviation of N_k, respectively. We set the initial weight of each point $\mathbf{x}$ in the training set to

$$w_k(\mathbf{x}) = \frac{1}{1+|h(\mathbf{x},k)|},$$

normalized by the sum over all points, for an empirically determined value of k. The motivation behind the weighting scheme is to assign less importance to both hubs and outliers than other points (this is why we take the absolute value of $h(\mathbf{x},k)$).

Figure 13 illustrates on six classification problems from Table 1 how the weighting scheme helps AdaBoost achieve better generalization in fewer iterations. The data sets were split into training,

24. More precisely, we use the binary "Real AdaBoost" algorithm and the one-vs-all scheme to handle multi-class problems, which is equivalent to the original AdaBoost.MH (Friedman et al., 2000).

validation, and test sets with size ratio 2:1:1, parameter k was chosen based on classification accuracy on the validation sets, and accuracies on the test sets are reported. While it is known that AdaBoost is sensitive to outliers (Rätsch et al., 2001), improved accuracy suggests that hubs should be regarded in an analogous manner, that is, both hubs and antihubs are intrinsically more difficult to classify correctly, and the attention of the weak learners should initially be focused on "regular" points. The discussion from Section 4.4, about hubs corresponding to probabilistic outliers in high-dimensional data, offers an explanation for the observed good performance of the weighting scheme, as both hubs and antihubs can be regarded as (probabilistic) outliers.

To provide further support, Figure 14 depicts binned accuracies of unweighted AdaBoost trained in one fifth of the iterations shown in Figure 13, for points sorted by decreasing N_k. It illustrates how in earlier phases of ensemble training the generalization power with hubs and/or antihubs is worse than with regular points. Moreover, for the considered data sets it is actually the hubs that appear to cause more problems for AdaBoost than antihubs (that is, distance-based outliers).

7.2 Semi-Supervised Learning

Semi-supervised learning algorithms make use of data distribution information provided by unlabeled examples during the process of building a classifier model. An important family of approaches are graph-based methods, which represent data as nodes of a graph, the edges of which are weighted by pairwise distances of incident nodes (Chapelle et al., 2006).

We consider the well-known algorithm by Zhu et al. (2003), whose strategy involves computing a real-valued function f on graph nodes, and assign labels to nodes based on its values. Function f, which exhibits harmonic properties, is obtained by optimizing a quadratic energy function that involves graph edge weights, with the probability distribution on the space of functions f formed using Gaussian fields. For data points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ we consider edge weights assigned by the radial basis function (RBF) of the following form:

$$W(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2}\right),$$

where σ is a data-dependent constant. Therefore, large edge weights are assigned between nodes that are close to one another with respect to Euclidean distance.

Taking into account the properties of hubs and antihubs discussed in previous sections, for high-dimensional data sets it can be expected of hubs to be closer to many other points than "regular" points are, and thus carry larger edge weights and be more influential in the process of determining the optimal function f. Conversely, antihubs are positioned farther away from other points, and are expected to bear less influence on the computation of f. Therefore, following an approach that resembles active learning, selecting the initial labeled point set from hubs could be more beneficial in terms of classification accuracy than arbitrarily selecting the initial points to be labeled. On the other extreme, picking the initial labeled point set from the ranks of antihubs could be expected to have a detrimental effect on accuracy.

To validate the above hypothesis, we evaluated the accuracy of the harmonic function algorithm by Zhu et al. (2003) on multiple high-dimensional data sets from Table 1, for labeled set sizes ranging from 1% to 10% of the original data set size, with the test set consisting of all remaining unlabeled points. Because the setting is semi-supervised, we compute the N_k scores of points based on complete data sets, instead of only training sets which was the case in Section 7.1.2. Based on

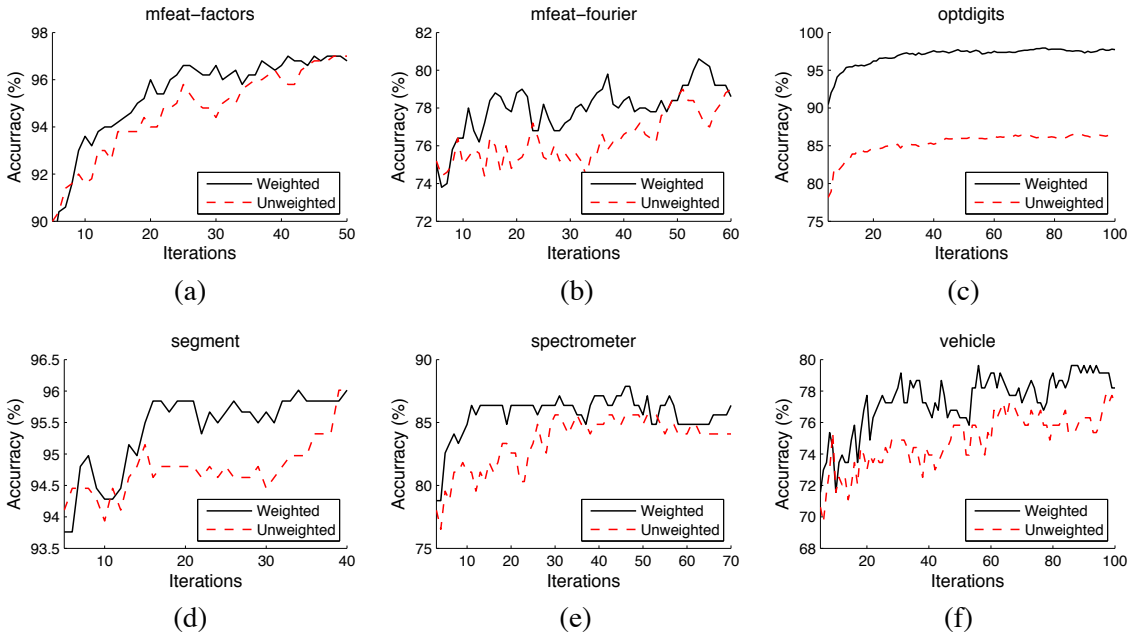


Figure 13: Accuracy of AdaBoost with and without the weighting scheme: (a) $k = 20$, (b) $k = 15$, (c) $k = 10$, (d) $k = 20$, (e) $k = 20$, (f) $k = 40$.

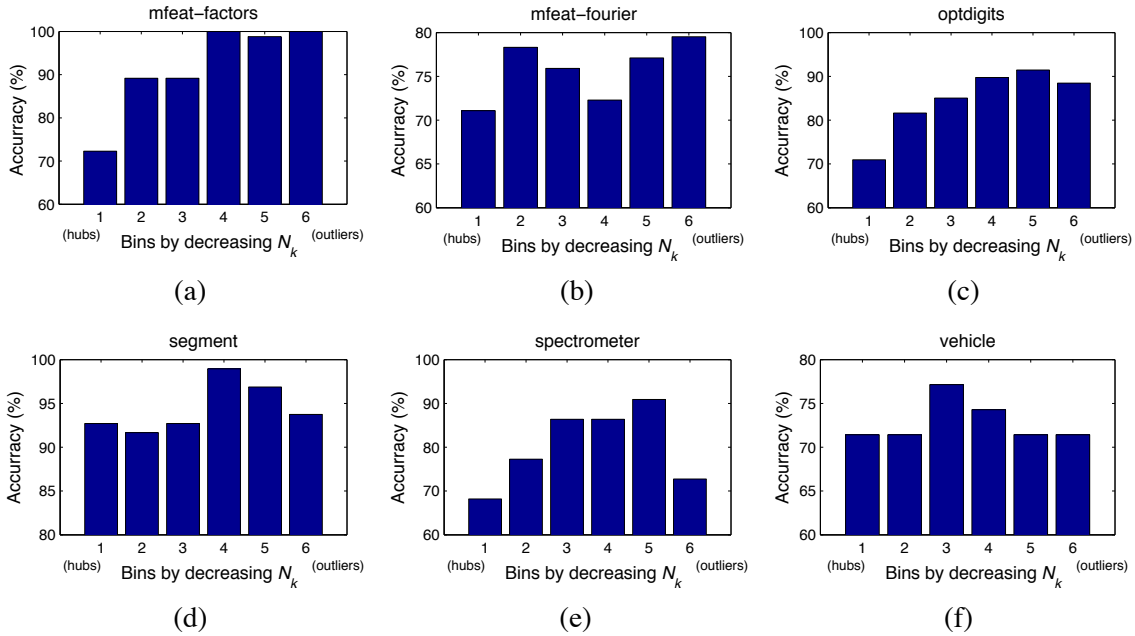


Figure 14: Binned accuracy of AdaBoost by decreasing N_k, at one fifth of the iterations shown in Figure 13.

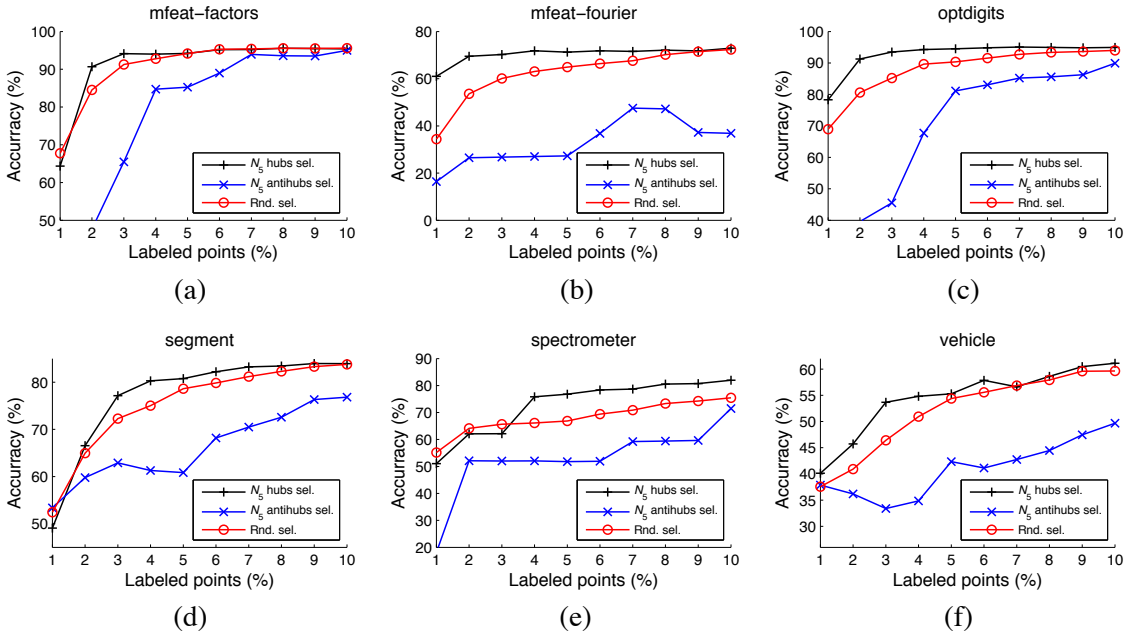


Figure 15: Accuracy of the semi-supervised algorithm by Zhu et al. (2003) with respect to the initial labeled set size as a percentage of the original data set size. Labeled points are selected in the order of decreasing N_5 (hubs first), increasing N_5 (antihubs first), and in random order. Sigma values of the RBF are: (a) $\sigma = 2.9$, (b) $\sigma = 2$, (c) $\sigma = 2.1$, (d) $\sigma = 0.9$, (e) $\sigma = 1.8$, and (f) $\sigma = 0.7$.

the N_k scores ($k = 5$), Figure 15 plots classification accuracies for labeled points selected in the order of decreasing N_5 (we choose to take hub labels first), in the order of increasing N_5 (antihub labels are taken first), and in random order (where we report accuracies averaged over 10 runs).[25] It can be seen that when the number of labeled points is low in comparison to the size of the data sets, taking hub labels first generally produces better classification accuracy. On the other hand, when assigning initial labels to antihubs, accuracy becomes significantly worse, with a much larger labeled set size required for the accuracy to reach that of randomly selected labeled points.

7.3 Unsupervised Learning

This section will discuss the interaction of the hubness phenomenon with unsupervised learning, specifically the tasks of clustering (Section 7.3.1) and outlier detection (Section 7.3.2).

7.3.1 CLUSTERING

The main objectives of (distance-based) clustering algorithms are to minimize intra-cluster distance and maximize inter-cluster distance. The skewness of k-occurrences in high-dimensional data influences both objectives.

25. We determined the σ values of the RBF function in a separate experiment involving 10 runs of random selection of points for the labeled set, the size of which is 10% of the original data set.

Intra-cluster distance may be increased due to points with low k-occurrences. As discussed in Section 4.4, such points are far from all the rest, acting as distance-based outliers. Distance-based outliers and their influence on clustering are well-studied subjects (Tan et al., 2005): outliers do not cluster well because they have high intra-cluster distance, thus they are often discovered and eliminated beforehand. The existence of outliers is attributed to various reasons (for example, erroneous measurements). Nevertheless, the skewness of N_k suggests that in high-dimensional data outliers are also expected due to inherent properties of vector space. Section 7.3.2 will provide further discussion on this point.

Inter-cluster distance, on the other hand, may be reduced due to points with high k-occurrences, that is, hubs. Like outliers, hubs do not cluster well, but for a different reason: they have low inter-cluster distance, because they are close to many points, thus also to points from other clusters. In contrast to outliers, the influence of hubs on clustering has not attracted significant attention.

To examine the influence of both outliers and hubs, we used the popular silhouette coefficients (SC) (Tan et al., 2005). For the ith point, let a_i be the average distance to all points in its cluster (a_i corresponds to intra-cluster distance), and b_i the minimum average distance to points from other clusters (b_i corresponds to inter-cluster distance). The SC of the ith point is $(b_i - a_i)/\max(a_i, b_i)$, ranging between -1 and 1 (higher values are preferred). The SC of a set of points is obtained by averaging the silhouette coefficients of the individual points.

We examined several clustering algorithms, and report results for the spectral algorithm of Meilă and Shi (2001) and Euclidean distance, with similar results obtained for classical K-means, as well as the spectral clustering algorithm by Ng et al. (2002) in conjunction with K-means and the algorithm by Meilă and Shi (2001). For a given data set, we set the number of clusters, K, to the number of classes (specified in Table 1). We select as hubs those points $\mathbf{x}$ with $h(\mathbf{x},k) > 2$, that is, $N_k(\mathbf{x})$ more than two standard deviations higher than the mean (note that $h(\mathbf{x},k)$, defined by Equation 24, ignores labels). Let n_h be the number of hubs selected. Next, we select as outliers the n_h points with the lowest k-occurrences. Finally, we randomly select n_h points from the remaining points (we report averages for 100 different selections). To compare hubs and antihubs against random points, we measure the *relative SC* of hubs (antihubs): the mean SC of hubs (antihubs) divided by the mean SC of random points. For several data sets from Table 1, Figure 16 depicts with bars the relative silhouette coefficients.[26] As expected, outliers have relative SC lower than one, meaning that they cluster worse than random points. Notably, the same holds for hubs, too.[27]

To gain further insight, Figure 16 plots with lines (referring to the right vertical axes) the relative mean values of a_i and b_i for hubs and outliers (dividing with those of randomly selected points). Outliers have high relative a_i values, indicating higher intra-cluster distance. Hubs, in contrast, have low relative b_i values, indicating reduced inter-cluster distance. In conclusion, when clustering high-dimensional data, hubs should receive analogous attention as outliers.

7.3.2 OUTLIER DETECTION

This section will briefly discuss possible implications of high dimensionality on distance-based outlier detection, in light of the findings concerning the hubness phenomenon presented in previous sections. Section 4.4 already discussed the correspondence of antihubs with distance-based outliers

26. Data sets were selected due to their high (intrinsic) dimensionality—similar observations can be made with other high-dimensional data sets from Table 1.
27. The statistical significance of differences between the SC of hubs and randomly selected points has been verified with the paired t-test at 0.05 significance level.

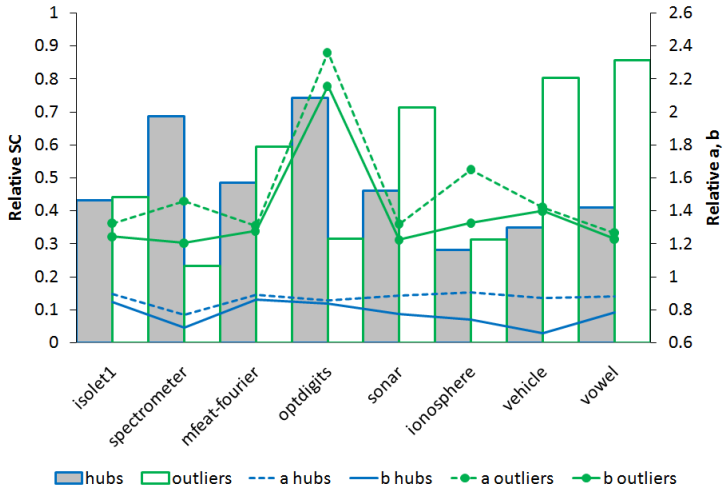


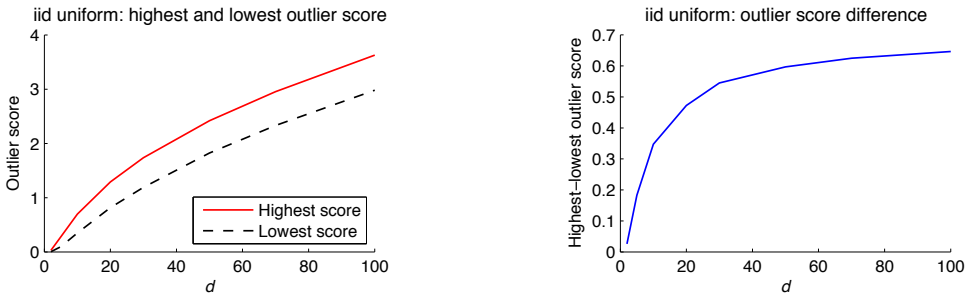
Figure 16: Relative silhouette coefficients for hubs (gray filled bars) and outliers (empty bars). Relative values for a and b coefficients are also plotted (referring to the right vertical axes).

in high dimensions, and demonstrated on real data the negative correlation between N_k and a commonly used outlier score—distance to the kth nearest neighbor. On the other hand, a prevailing view of the effect of high dimensionality on distance-based outlier detection is that, due to distance concentration, every point seems to be an equally good outlier, thus hindering outlier-detection algorithms (Aggarwal and Yu, 2001). Based on the observations regarding hubness and the behavior of distances discussed earlier, we believe that the true problem actually lies in the opposite extreme: high dimensionality induces antihubs that can represent "artificial" outliers. This is because, from the point of view of common distance-based outlier scoring schemes, antihubs may appear to be stronger outliers in high dimensions than in low dimensions, only due to the effects of increasing dimensionality of data.

To illustrate the above discussion, Figure 17(a) plots for i.i.d. uniform random data the highest and lowest outlier score (distance to the kth NN, $k = 5$) with respect to increasing d ($n = 10000$ points, averages over 10 runs are reported). In accordance with the asymptotic behavior of all pairwise distances discussed in previous sections, both scores increase with d. However, as Figure 17(b) shows, the *difference* between the two scores also increases. This implies that a point could be considered a distance-based outlier only because of high dimensionality, since outliers are not expected for any other reason in i.i.d. uniform data (we already demonstrated that such a point would most likely be an antihub). As a consequence, outlier-detection methods based on measuring distances between points may need to be adjusted to account for the (intrinsic) dimensionality of data, in order to prevent dimensionality-induced false positives.

8. Conclusion

In this paper we explored an aspect of the curse of dimensionality that is manifested through the phenomenon of hubness—the tendency of high-dimensional data sets to contain hubs, in the sense of popular nearest neighbors of other points. To the best of our knowledge, hubs and the effects they



(a) (b)

Figure 17: (a) Highest and lowest distance to the 5th nearest neighbor in i.i.d. uniform random data, with respect to increasing d. (b) The difference between the two distances.

have on machine-learning techniques have not been thoroughly studied subjects. Through theoretical and empirical analysis involving synthetic and real data sets we demonstrated the emergence of the phenomenon and explained its origins, showing that it is an inherent property of data distributions in high-dimensional vector space that depends on the intrinsic, rather than embedding dimensionality of data. We also discussed the interaction of hubness with dimensionality reduction. Moreover, we explored the impact of hubness on a wide range of machine-learning tasks that directly or indirectly make use of distances between points, belonging to supervised, semi-supervised, and unsupervised learning families, demonstrating the need to take hubness into account in an equivalent degree to other factors, like the existence of outliers.

Besides application areas that involve audio and image data (Aucouturier and Pachet, 2007; Doddington et al., 1998; Berenzweig, 2007; Hicklin et al., 2005), identifying hubness within data and methods from other fields can be considered an important aspect of future work, as well as designing application-specific methods to mitigate or take advantage of the phenomenon. We already established the existence of hubness and its dependence on data dimensionality on collaborative filtering data with commonly used variants of cosine distance (Nanopoulos et al., 2009), time-series data sets in the context of k-NN classification involving dynamic time warping (DTW) distance (Radovanović et al., 2010b), text data within several variations of the classical vector space model for information retrieval (Radovanović et al., 2010a), and audio data for music information retrieval using spectral similarity measures (Karydis et al., 2010). In the immediate future we plan to perform a more detailed investigation of hubness in the fields of outlier detection and image mining. Another application area that could directly benefit from an investigation into hubness are reverse k-NN queries (which retrieve data points that have the query point $\mathbf{q}$ as one of their k nearest neighbors, Tao et al., 2007).

One concern we elected not to include into the scope of this paper is the efficiency of computing N_k. It would be interesting to explore the interaction between approximate k-NN graphs (Chen et al., 2009) and hubness, in both directions: to what degree do approximate k-NN graphs preserve hubness information, and can hubness information be used to enhance the computation of approximate k-NN graphs for high-dimensional data (in terms of both speed and accuracy).

Possible directions for future work within different aspects of machine learning include a more formal and theoretical study of the interplay between hubness and various distance-based machine-

learning models, possibly leading to approaches that account for the phenomenon at a deeper level. Supervised learning methods may deserve special attention, as it was also observed in another study (Caruana et al., 2008) that the k-NN classifier and boosted decision trees can experience problems in high dimensions. Further directions of research may involve determining whether the phenomenon is applicable to probabilistic models, (unboosted) decision trees, and other techniques not explicitly based on distances between points; and also to algorithms that operate within general metric spaces. Since we determined that for K-means clustering of high-dimensional data hubs tend to be close to cluster centers, it would be interesting to explore whether this can be used to improve iterative clustering algorithms, like K-means or self-organizing maps (Kohonen, 2001). Nearest-neighbor clustering (Bubeck and von Luxburg, 2009) of high-dimensional data may also directly benefit from hubness information. Topics that could also be worth further study are the interplay of hubness with learned metrics (Weinberger and Saul, 2009) and dimensionality reduction, including supervised (Vlassis et al., 2002; Geng et al., 2005), semi-supervised (Zhang et al., 2007), and unsupervised approaches (van der Maaten et al., 2009; Kumar, 2009). Finally, as we determined high correlation between intrinsic dimensionality and the skewness of N_k, it would be interesting to see whether some measure of skewness of the distribution of N_k can be used for estimation of the intrinsic dimensionality of a data set.

Acknowledgments

We would like to thank the authors of all software packages, libraries, and resources used in this research: Matlab with the Statistics Toolbox, Dimensionality Reduction Toolbox (van der Maaten, 2007), the FastICA package (Gävert et al., 2005), GML AdaBoost Toolbox (Vezhnevets, 2006), code for semi-supervised learning with the basic harmonic function (Zhu et al., 2003), SpectraLIB package for symmetric spectral clustering (Verma, 2003; Meilă and Shi, 2001), MatlabBGL graph library (Gleich, 2008); the Weka machine-learning toolkit (Witten and Frank, 2005), and Wolfram Mathematica with invaluable online resources MathWorld (mathworld.wolfram.com/) and the Wolfram Functions Site (functions.wolfram.com/).

We would like to express our gratitude to the action editor, Ulrike von Luxburg, and the anonymous reviewers, for helping to significantly improve the article with their detailed and thoughtful comments. We are also indebted to Zagorka Lozanov-Crvenković for helpful discussions and suggestions regarding the mathematical aspects of the paper.

Miloš Radovanović and Mirjana Ivanović thank the Serbian Ministry of Science for support through project *Abstract Methods and Applications in Computer Science*, no. 144017A. Alexandros Nanopoulos gratefully acknowledges the partial co-funding of his work through the European Commission FP7 project *MyMedia* (www.mymediaproject.org/) under the grant agreement no. 215006.

References

Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions With Formulas, Graphs and Mathematical Tables*. National Bureau of Standards, USA, 1964.

Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *Proceedings of the 27th ACM SIGMOD International Conference on Management of Data*, pages 37–46, 2001.

Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Proceedings of the 8th International Conference on Database Theory (ICDT)*, volume 1973 of *Lecture Notes in Computer Science*, pages 420–434. Springer, 2001.

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

Jean-Julien Aucouturier and Francois Pachet. A scale-free distribution of false positives for a large class of audio similarity measures. *Pattern Recognition*, 41(1):272–284, 2007.

Richard E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

Adam Berenzweig. *Anchors and Hubs in Audio-based Music Similarity*. PhD thesis, Columbia University, New York, USA, 2007.

Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT)*, volume 1540 of *Lecture Notes in Computer Science*, pages 217–235. Springer, 1999.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Chapman & Hall, 1984.

Sébastien Bubeck and Ulrike von Luxburg. Nearest neighbor clustering: A baseline method for consistent clustering with arbitrary objective functions. *Journal of Machine Learning Research*, 10:657–698, 2009.

Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 96–103, 2008.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006.

Jie Chen, Haw ren Fang, and Yousef Saad. Fast approximate *k*NN graph construction for high dimensional data via recursive Lanczos bisection. *Journal of Machine Learning Research*, 10: 1989–2012, 2009.

Pierre Demartines. *Analyse de Données par Réseaux de Neurones Auto-Organisés*. PhD thesis, Institut national polytechnique de Grenoble, Grenoble, France, 1994.

George Doddington, Walter Liggett, Alvin Martin, Mark Przybocki, and Douglas Reynolds. SHEEP, GOATS, LAMBS and WOLVES: A statistical analysis of speaker performance in the NIST 1998 speaker recognition evaluation. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP)*, 1998. Paper 0608.

Robert J. Durrant and Ata Kabán. When is 'nearest neighbour' meaningful: A converse theorem and implications. *Journal of Complexity*, 25(4):385–397, 2009.

Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

Paul F. Evangelista, Mark J. Embrechts, and Boleslaw K. Szymanski. Taming the curse of dimensionality in kernels and novelty detection. In A. Abraham, B. Baets, M. Koppen, and B. Nickolay, editors, *Applied Soft Computing Technologies: The Challenge of Complexity*, volume 34 of *Advances in Soft Computing*, pages 425–438. Springer, 2006.

Damien François. *High-dimensional Data Analysis: Optimal Metrics and Feature Selection*. PhD thesis, Université catholique de Louvain, Louvain, Belgium, 2007.

Damien François, Vincent Wertz, and Michel Verleysen. The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):873–886, 2007.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000.

Yasunori Fujikoshi. Computable error bounds for asymptotic expansions of the hypergeometric function ${}_1F_1$ of matrix argument and their applications. *Hiroshima Mathematical Journal*, 37(1): 13–23, 2007.

Hugo Gävert, Jarmo Hurri, Jaakko Särelä, and Aapo Hyvärinen. The FastICA package for Matlab. http://www.cis.hut.fi/projects/ica/fastica/, 2005.

Xin Geng, De-Chuan Zhan, and Zhi-Hua Zhou. Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(6):1098–1107, 2005.

David Gleich. MatlabBGL: A Matlab graph library. http://www.stanford.edu/∼dgleich/programs/matlab_bgl/, 2008.

Uffe Haagerup. The best constants in the Khintchine inequality. *Studia Mathematica*, 70:231–283, 1982.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

Austin Hicklin, Craig Watson, and Brad Ulery. The myth of goats: How many people have fingerprints that are hard to match? Internal Report 7271, National Institute of Standards and Technology (NIST), USA, 2005.

Alexander Hinneburg, Charu C. Aggarwal, and Daniel A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB)*, pages 506–515, 2000.

Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pages 833–840, 2003.

Chih-Ming Hsu and Ming-Syan Chen. On the design and applicability of distance functions in high-dimensional data space. *IEEE Transactions on Knowledge and Data Engineering*, 21(4): 523–536, 2009.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4–5):411–430, 2000.

Khandoker Tarik-Ul Islam, Kamrul Hasan, Young-Koo Lee, and Sungyoung Lee. Enhanced 1-NN time series classification using badness of records. In *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*, pages 108–113, 2008.

Kiyosi Itô, editor. *Encyclopedic Dictionary of Mathematics*. The MIT Press, 2nd edition, 1993.

Tony Jebara, Jun Wang, and Shih-Fu Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 441–448, 2009.

Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 1. Wiley, 2nd edition, 1994.

Ian T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

Ioannis Karydis, Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Looking through the "glass ceiling": A conceptual framework for the problems of spectral similarity. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010.

S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and K. R. Krishna Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13(3): 637–649, 2001.

Teuvo Kohonen. *Self-Organizing Maps*. Springer, 3rd edition, 2001.

Filip Korn, Bernd-Uwe Pagel, and Christos Faloutsos. On the "dimensionality curse" and the "self-similariy blessing". *IEEE Transactions on Knowledge and Data Engineering*, 13(1):96–111, 2001.

Ch. Aswani Kumar. Analysis of unsupervised dimensionality reduction techniques. *Computer Science and Information Systems*, 6(2):217–227, 2009.

Stéphane Lafon and Ann B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006.

Elizaveta Levina and Peter J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems 17*, pages 777–784, 2005.

Lun Li, David Alderson, John C. Doyle, and Walter Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4):431–523, 2005.

Laurence T. Maloney. Nearest neighbor analysis of point processes: Simulations and evaluations. *Journal of Mathematical Psychology*, 27(3):251–260, 1983.

Marina Meilă and Jianbo Shi. Learning segmentation by random walks. In *Advances in Neural Information Processing Systems 13*, pages 873–879, 2001.

Boaz Nadler, Stéphane Lafon, Ronald R. Coifman, and Ioannis G. Kevrekidis. Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis*, 21(1):113–127, 2006.

Alexandros Nanopoulos, Miloš Radovanović, and Mirjana Ivanović. How does high dimensionality affect collaborative filtering? In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys)*, pages 293–296, 2009.

Charles M. Newman and Yosef Rinott. Nearest neighbors and Voronoi volumes in high-dimensional point processes with various distance functions. *Advances in Applied Probability*, 17(4):794–809, 1985.

Charles M. Newman, Yosef Rinott, and Amos Tversky. Nearest neighbors and Voronoi regions in certain point processes. *Advances in Applied Probability*, 15(4):726–751, 1983.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2002.

Luca Oberto and Francesca Pennecchi. Estimation of the modulus of a complex-valued quantity. *Metrologia*, 43(6):531–538, 2006.

Andrew M. Odlyzko and Neil J. A. Sloane. New bounds on the number of unit spheres that can touch a unit sphere in *n* dimensions. *Journal of Combinatorial Theory, Series A*, 26(2):210–214, 1979.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–278, 2004.

Mathew Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, pages 185–208. MIT Press, 1999.

Miloš Radovanović and Mirjana Ivanović. Document representations for classification of short Web-page descriptions. In *Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, volume 4081 of *Lecture Notes in Computer Science*, pages 544–553. Springer, 2006.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Nearest neighbors in high-dimensional data: The emergence and influence of hubs. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 865–872, 2009.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. On the existence of obstinate results in vector space models. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 186–193, 2010a.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Time-series classification in many intrinsic dimensions. In *Proceedings of the 10th SIAM International Conference on Data Mining (SDM)*, pages 677–688, 2010b.

Gunnar Rätsch, Takashi Onoda, and Klaus-Robert Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

John P. Scott. *Social Network Analysis: A Handbook*. Sage Publications, 2nd edition, 2000.

Amit Singh, Hakan Ferhatosmanoğlu, and Ali Şaman Tosun. High dimensional reverse nearest neighbor queries. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM)*, pages 91–98, 2003.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2005.

Yufei Tao, Dimitris Papadias, Xiang Lian, and Xiaokui Xiao. Multidimensional reverse *k*NN search. *VLDB Journal*, 16(3):293–316, 2007.

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Amos Tversky and John Wesley Hutchinson. Nearest neighbor analysis of psychological spaces. *Psychological Review*, 93(1):3–22, 1986.

Amos Tversky, Yosef Rinott, and Charles M. Newman. Nearest neighbor analysis of point processes: Applications to multidimensional scaling. *Journal of Mathematical Psychology*, 27(3): 235–250, 1983.

Laurens van der Maaten. An introduction to dimensionality reduction using Matlab. Technical Report MICC 07-07, Maastricht University, Maastricht, The Netherlands, 2007.

Laurens van der Maaten, Eric Postma, and Jaap van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, Tilburg, The Netherlands, 2009.

Deepak Verma. SpectraLIB – package for symmetric spectral clustering. http://www.stat.washington.edu/spectral/, 2003.

Alexander Vezhnevets. GML AdaBoost Matlab Toolbox. MSU Graphics & Media Lab, Computer Vision Group, http://graphics.cs.msu.ru/, 2006.

Nikos Vlassis, Yoichi Motomura, and Ben Kröse. Supervised dimension reduction of intrinsically low-dimensional data. *Neural Computation*, 14(1):191–215, 2002.

Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2nd edition, 2005.

Yi-Ching Yao and Gordon Simons. A large-dimensional independent and identically distributed property for nearest neighbor counts in Poisson processes. *Annals of Applied Probability*, 6(2): 561–571, 1996.

Kenneth Zeger and Allen Gersho. Number of nearest neighbors in a Euclidean code. *IEEE Transactions on Information Theory*, 40(5):1647–1649, 1994.

Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Semi-supervised dimensionality reduction. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM)*, pages 629–634, 2007.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 912–919, 2003.

Journal of Machine Learning Research 11 (2010) 2533-2541 Submitted 6/10; Revised 8/10; Published 9/10

WEKA—Experiences with a Java Open-Source Project

Remco R. Bouckaert REMCO@CS.WAIKATO.AC.NZ
Eibe Frank EIBE@CS.WAIKATO.AC.NZ
Mark A. Hall MHALL@CS.WAIKATO.AC.NZ
Geoffrey Holmes GEOFF@CS.WAIKATO.AC.NZ
Bernhard Pfahringer BERNHARD@CS.WAIKATO.AC.NZ
Peter Reutemann FRACPETE@CS.WAIKATO.AC.NZ
Ian H. Witten IHW@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikatoi
Hamilton, New Zealand

Editor: Soeren Sonnenburg

Abstract

WEKA is a popular machine learning workbench with a development life of nearly two decades. This article provides an overview of the factors that we believe to be important to its success. Rather than focussing on the software's functionality, we review aspects of project management and historical development decisions that likely had an impact on the uptake of the project.

Keywords: machine learning software, open source software

1. Introduction

We present a brief account of the WEKA 3 software, which is distributed under the GNU General Public License, followed by some lessons learned over the period spanning its development and maintenance. We also include a brief historical mention of its predecessors.

WEKA contains implementations of algorithms for classification, clustering, and association rule mining, along with graphical user interfaces and visualization utilities for data exploration and algorithm evaluation. This article shares some background on software design and management decisions, in the hope that it may prove useful to others involved in the development of open-source machine learning software. Hall et al. (2009) give an overview of the system; more comprehensive sources of information are Witten and Frank's book *Data Mining* (2005) and the user manuals included in the software distribution.[1] Online sources, including the WEKA Wiki pages[2] and the API, provide the most complete coverage. The *wekalist* mailing list is a forum for discussion of WEKA-related queries, with nearly 3000 subscribers.

2. What is WEKA?

WEKA is a machine learning workbench that supports many activities of machine learning practitioners.

1. Available from `http://www.cs.waikato.ac.nz/ml/weka`.
2. Available at `http://weka.wikispaces.com/`.

2.1 Basic Functionality

Here is a summary of WEKA's main features.

- *Data preprocessing*. As well as a native file format (ARFF), WEKA supports various other formats (for instance CSV, Matlab ASCII files), and database connectivity through JDBC. Data can be filtered by a large number of methods (over 75), ranging from removing particular attributes to advanced operations such as principal component analysis.

- *Classification*. One of WEKA's drawing cards is the more than 100 classification methods it contains. Classifiers are divided into "Bayesian" methods (Naive Bayes, Bayesian nets, etc.), lazy methods (nearest neighbor and variants), rule-based methods (decision tables, OneR, RIPPER), tree learners (C4.5, Naive Bayes trees, M5), function-based learners (linear regression, SVMs, Gaussian processes), and miscellaneous methods. Furthermore, WEKA includes meta-classifiers like bagging, boosting, stacking; multiple instance classifiers; and interfaces for classifiers implemented in Groovy and Jython.

- *Clustering*. Unsupervised learning is supported by several clustering schemes, including EM-based mixture models, k-means, and various hierarchical clustering algorithms. Though not as many methods are available as for classification, most of the classic algorithms are included.

- *Attribute selection*. The set of attributes used is essential for classification performance. Various selection criteria and search methods are available.

- *Data visualization*. Data can be inspected visually by plotting attribute values against the class, or against other attribute values. Classifier output can be compared to training data in order to detect outliers and observe classifier characteristics and decision boundaries. For specific methods there are specialized tools for visualization, such as a tree viewer for any method that produces classification trees, a Bayes network viewer with automatic layout, and a dendrogram viewer for hierarchical clustering.

WEKA also includes support for association rule mining, comparing classifiers, data set generation, facilities for annotated documentation generation for source code, distribution estimation, and data conversion.

2.2 Graphical User Interfaces

WEKA's functionality can be accessed through various graphical user interfaces, principally the Explorer, and Experimenter interfaces shown in Figure 1, but also the Knowledge Flow interface. The most popular interface, the Explorer, allows quick exploration of data and supports all the main items mentioned above—data loading and filtering, classification, clustering, attribute selection and various forms of visualization—in an interactive fashion.

The Experimenter is a tool for setting up machine learning experiments that evaluate classification and regression methods. It allows easy comparison of performance, and can tabulate summaries in ways that are easy to incorporate into publications. Experiments can be set up to run in parallel over different computers in a network so that multiple repetitions of cross validation (the default method of performance analysis) can be distributed over multiple machines.

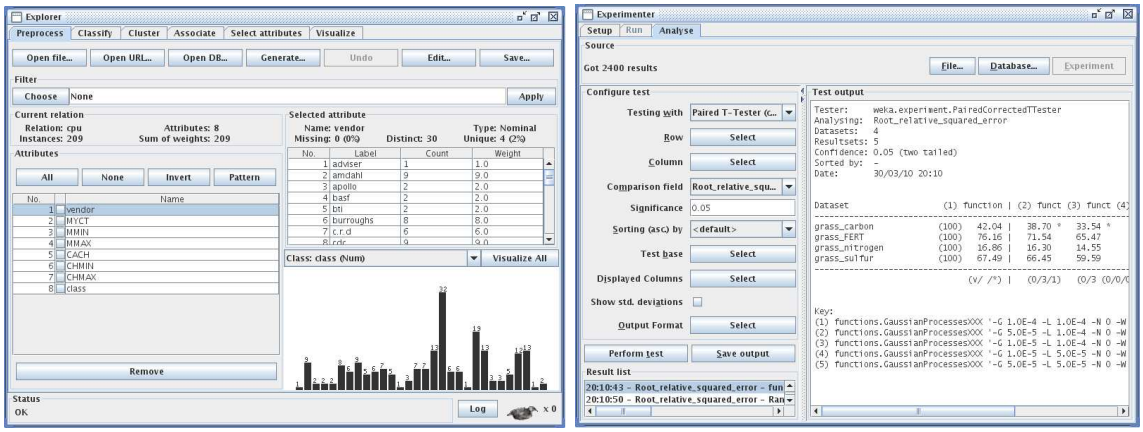


Figure 1: The Explorer and Experimenter interfaces.

The Knowledge Flow interface is a Java Beans application that allows the same kind of data exploration, processing and visualization as the Explorer (along with some extras), but in a workflow-oriented system. The user can define a workflow specifying how data is loaded, preprocessed, evaluated and visualized, which can be repeated multiple times. This makes it easy to optimize the workflow by tweaking parameters of algorithms, or to apply it to other data sources. In the Explorer, on the other hand, the individual steps must be invoked manually, one at a time. This is a rather tedious process, and is prone to errors such as omitting preprocessing steps.

WEKA also includes some specialized graphical interfaces, such as a Bayes network editor that focuses on Bayes network learning and inference, an SQL viewer for interaction with databases, and an ARFF data file viewer and editor.

All functionality and some more specialized functions can be accessed from a command line interface, so WEKA can be used without a windowing system.

2.3 Extending WEKA

One of WEKA's major strengths is that it is easily extended with customized or new classifiers, clusterers, attribute selection methods, and other components. For instance, all that is needed to add a new classifier is a class that derives from the `Classifier` class and implements the `buildClassifier` method for learning, and a `classifyInstance` method for testing/predicting the value for a data point. The code fragment in Figure 2 shows a minimal implementation of a classifier that returns the mean or mode of the class in the training set (double values are used to store indices of nominal attribute values).

Any new class is picked up by the graphical user interfaces through Java introspection: no further coding is needed to deploy it from WEKA's graphical user interfaces. This makes it easy to evaluate how new algorithms perform compared to any of the existing ones, which explains WEKA's popularity among machine learning researchers.

Besides being easy to extend, WEKA includes a wide range of support for adding functionality to basic implementations. For instance, a classifier can have various optional settings by implementing a pre-defined interface for option handling. Each option can be documented using a tool-tip text

```
package weka.classifiers.misc;

import weka.classifiers.Classifier;
import weka.core.*;

public class NewClassifier extends Classifier {
        double m_fMean;

        public void buildClassifier(Instances data) throws Exception {
                m_fMean = data.meanOrMode(data.classIndex());
        }

        public double classifyInstance(Instance instance) throws Exception {
                return m_fMean;
        }

}
```

Figure 2: Classifier code example.

method, which is picked up by the *help* dialogs of the graphical user interfaces. Classes typically implement methods described in papers, and this provenance metadata can be captured in a method that is used to generate documentation. Some methods only apply to certain kinds of data, such as numeric class values or discrete attribute values. A 'capabilities' mechanism allows classes to identify what kind of data is acceptable by any method, and the graphical user interfaces incorporate this by making methods available only if they are able to process the data at hand.

There are many projects that build on top WEKA, about fifty of which are listed on the WEKA Wiki.

3. Origins

The Machine Learning project at Waikato was launched in 1993 with a successful grant application to the New Zealand Foundation for Research, Science, and Technology. The underlying intention behind the request was not so much to further a specific agenda in machine learning research as to create a research culture in a small and obscure computer science department that brought different people together. Machine learning was selected because of prior expertise and its potential applicability to agriculture, New Zealand's core industry; the grant was justified in terms of applications research rather than the development of new learning techniques.

This gave the research team a license to incorporate and reimplement existing methods, and work soon began on a workbench, written in C, that was intended to provide a common interface to a growing collection of machine learning algorithms. It contained some learning algorithms written mostly in C, data I/O and pre-processing tools, also written in C, and graphical user interfaces written in TCL/TK. The number of learning algorithms was limited and they came from different sources; wrapper shell scripts were employed to bind them into the framework. The acronym WEKA for "Waikato Environment for Knowledge Analysis" was coined, and the system gradually became known in the international ML community, along with another machine learning library in C++ from the University of Stanford called MLC++, developed by Kohavi et al. (1997).[3]

3. MLC++ is now distributed by Silicon Graphics at `http://www.sgi.com/Technology/mlc`.

Because of dependencies on other libraries, mainly related to the graphical user interfaces, the software became increasingly unwieldy and hard to maintain. In 1997 work began on reimplementing WEKA from scratch in Java into what we now term WEKA 3. One of the authors, Eibe Frank, had earlier decided to adopt Java to implement algorithm prototypes, abandoning C++ because Java development was rapid and debugging was dramatically simplified. This positive experience, the promise of platform independence through virtual machine technology, and the fact that, as part of the research code, classes for reading WEKA's standard ARFF file format into appropriate data structures already existed, led to the decision to re-write WEKA in Java.

Many of the classes in today's code archive (including, for example, J48, the WEKA implementation of Quinlan's C4.5) date from mid-1997. Rapid early development was stimulated by the need to teach a course on machine learning at the University of Calgary during a Fall 1997 sabbatical visit by Witten, along with several students, including Frank. The Java version was called JAWS for "Java Weka system" to avoid confusion with WEKA itself, and after some debate was changed to WEKA in March 1999. The first paper written using this system was Frank and Witten (1998), written in late 1997. By the end of 1998 WEKA included packages for classifiers, association rule learners, filters, and evaluation, as well as a core package. Several methods that were relatively advanced for the day, such as bagging, were in place, as well as old chestnuts like instance-based learning and Naive Bayes. Attribute selection was added soon afterwards, in 1999.

Work began on the first edition of the *Data Mining* book in 1997, based on earlier notes for Witten's courses at the University of Waikato, and a proposal was submitted to Morgan Kaufmann late that year. It finally appeared in 1999 (though for reasons that are not clear to us the official publication date is 2000). WEKA was seen as an important adjunct to the book, and the original title, *Practical machine learning*, was changed to *Data Mining: Practical machine learning tools and techniques with Java* to reflect this. The term "data mining" was prepended primarily for marketing reasons. The WEKA software described in that edition was command-line oriented and the book makes no mention of a graphical user interface, for which design began in 1999. By the time the second edition appeared in 2005 the interactive versions of WEKA—the Explorer, Experimenter, and Knowledge Flow interface—were mature and well-tested pieces of software.

4. How Did WEKA Become Widely Adopted?

The size of the mailing list, the volume of downloads, and the number of academic papers citing WEKA-based results show that the software is widely deployed. It is used in the machine learning and data mining community as an educational tool for teaching both applications and technical internals of machine learning algorithms, and as a research tool for developing and empirically comparing new techniques. It is applied increasingly widely in other academic fields, and in commercial settings. We are often asked what is the secret of WEKA's success, and here we speculate on reasons for the software's broad uptake. An obvious one is that it is free and open-source software. However, there are several other factors, many of which are exposed by the above brief historical review.

4.1 Portability

Pre-Java versions of WEKA were limited to UNIX operating systems and distributions were made available for Linux, Solaris, and SGI. The present popularity of the software owes much to the existence of Java Virtual Machines for all important platforms, along with the fact that all code

necessary to compile and run WEKA is included in the distribution. The lack of dependency on externally-maintained libraries makes maintenance of the code base much easier.

The decision to rewrite WEKA from scratch in Java may seem obvious in hindsight, but was a large step into the unknown at the time. For one thing, portability seemed less important in the days when everyone we knew was using Unix! More importantly, in 1997 no-one could forsee that Java would turn out to be such a suitable platform. Just-in-time compilation was unknown (Sun's Java 1.2, which included a just-in-time compiler, was introduced in December 1998). Fully-interpreted execution suffered a substantial performance hit compared to C++. And deficiencies in early Java technology still affect the WEKA code today: the original design of the basic data structures was clean, object-oriented, and elegant, but the performance penalty incurred by extensive use of objects led to a less generic array-based representation, where all data is stored in arrays of doubles. Many of the design characteristics of WEKA that seem inelegant are due to pragmatic decisions based on the Java technology available at the time.

Later versions of the Java Virtual Machine virtually eliminated the performance gap from C++ through just-in-time compilation and adaptive optimization. Although there still persists a perception that execution of Java code is too slow for scientific computing, this is not our experience and does not appear to be shared by the WEKA community.

4.2 Graphical User Interface

Early releases of the WEKA 3 software were command-line driven and did not include graphical user interfaces. Although many experienced users still shun them, the graphical interfaces undoubtedly contributed to the popularity of WEKA. The introduction of the Explorer in particular has made the software more accessible to users who want to employ machine learning in practice. It has allowed many universities (including our own) to offer courses in applied machine learning and data mining, and has certainly contributed to WEKA's popularity for commercial and industrial use. Again, while obvious in hindsight, the development of graphical user interfaces was a significant risk, because valuable programming effort had to be diverted from the main job of implementing learning algorithms into relatively superficial areas of presentation and interaction.

The WEKA 3 graphical user interface development benefited from the fact that the core system was already up and running, and relatively mature—as evidenced by the first edition of *Data Mining*—before any work began on interactive interfaces. The early WEKA project probably suffered from attempting to develop interactive interfaces (in TCL/TK) at the same time as the basic algorithms and data structures were being commissioned, a mistake that was avoided in the later system.

4.3 The *Data Mining* Book

The first stable release of the WEKA 3 software coincided with the publication of the first edition of *Data Mining* by Witten and Frank, which contained a chapter describing it, both interactively via the command-line and programmatically via the API and extension of superclasses. There has been a symbiotic relationship between the software and the book: users of the software are likely to consult the book and readers of the book are likely to try out the software. The combination of a book explaining the core algorithms in a corresponding piece of free software is particularly suitable for education. It seems likely that the feedback loop between the readership of the book and the users of the software has bolstered the size of both populations. The early existence of a

companion book, unusual for open source software, is particularly valuable for machine learning because the techniques involved are quite simple and easy to explain, but by no means obvious.

4.4 Extensibility

WEKA can be used both for educational purposes and as a research platform. New algorithms can easily be incorporated and compared to existing ones on a collection of data sets. As noted earlier, only two methods from the `Classifier` superclass need be implemented to come up with a classifier that takes advantage of all the infrastructure in WEKA, including I/O, evaluation, and so-called "meta" algorithms.

4.5 Documentation

In recent years there has been a dramatic growth in the volume of WEKA-related online documentation—notably as part of the WEKA Wiki. This, in conjunction with the information in the mailing list archives, provides a wealth of information for all users. The existence of a steadily increasing, knowledgeable and enthusiastic user community, and the combined knowledge they share, has played a significant role in the success of the software.

4.6 Comprehensiveness

Perhaps the foremost reason for the adoption of WEKA 3 by the research community has been the inclusion of faithful reimplementations of classic benchmark methods, in particular the C4.5 decision tree learner (Quinlan, 1992), but also other algorithms such as the RIPPER rule learner (Cohen, 1995). The original implementations of these algorithms were already very successful software projects in themselves. Bringing them together in a common framework has been a strong drawing card.

4.7 Support

"Given sufficient funding, anyone could have done that!" is a refrain often heard from sceptics. We were lucky to receive an initial research grant for *applied* machine learning research from a New Zealand funding agency that approved of our aspirations to investigate the application of this technology in agricultural domains. Yet it is hard to reconcile the practical need to win academic credit for research publications with the production of usable software, particularly when there is a constantly growing pressure to commercialize. We continued to apply for, and receive, follow-on funding from the same source, but—particularly as time went on—this compelled us to channel much of our research in the direction of target applications rather than basic research in machine learning.

5. Maintaining the Project

A software project can only become and remain successful if it is consistently maintained to a high standard. It has been our experience that this requires a group of people who are continually involved in the management and development of the software for an extended period of time, spanning several years. The core development team of WEKA has always been small and close-knit: having a small team helps maintain code quality and overall coherence.

Development of the software would not have been possible without financial support from the New Zealand government in the form of successive research grants by the Foundation for Research, Science and Technology, which have been awarded over a significant period of time—from 1993 to 2007. Over the years, work on the project has been done by a couple of academic staff, who were involved in the longer term and fitted it in with their teaching and research duties, and a succession of one (or one and a half) full-time equivalent research programmers. A fair amount of work was undertaken by students on casual contracts or as part of their studies. The community also contributed many algorithm implementations that are included in the WEKA distribution, along with some patches for the basic framework.

The project has always had a policy of maintaining close control of what became part of the software. Only a handful of developers have ever had write access to the source code repository. The drawback of this policy is reduced functionality; the advantages are improved code quality, ease of maintenance, and greater coherence for both developer and end user. When new algorithm implementations were considered for inclusion, we generally insisted on a backing publication describing the new method. In a few cases we have rejected submitted code, despite publication, when our experiments revealed that the method did not appear to improve on what was already present in WEKA.

The research contract that sponsored WEKA development required some measure of commercialization, and a few commercial licenses to parts of the WEKA code base owned by the University of Waikato have been sold. It eventually became clear that the succession of research contracts had a finite life span, and support by a commercial organization was necessary to keep WEKA healthy. Since 2007, Pentaho Corporation, a company that provides open-source business intelligence software and support, has contributed substantially to the maintenance of WEKA by hiring one of the chief developers and providing online help. As part of the requirement to commercialize the software, it has been necessary to maintain a branch in the source code repository that only contains code owned by the University of Waikato, an onerous but necessary facet of project maintenance.

6. Concluding Remarks

Obviously, in almost two decades of project development, many mistakes were made—but most were quickly corrected. One, mentioned above, regards the premature design of interactive interfaces, where WEKA 3 benefited from a strategic error made in the early WEKA project. Below are two instances of how adoption might have been strengthened had the project been managed differently.

One of the most challenging aspects of managing open source software development is to decide what to include in the software. Although careful control of contributions has substantial benefits, it limits community involvement (Bacon, 2009). A package-based architecture (such as that adopted by the R Development Team, 2009) provides a better platform for more widespread ownership and development. Under such a scheme, packages maintained by their developers can be loaded into the system on demand, opening it up to greater diversity and flexibility. A recent development in WEKA is the inclusion of package management, so that packages can easily be added to a given installation.[4] The project would probably have benefited by moving in this direction earlier.

We have learned that mailing lists for open source software are easier to maintain if the users are researchers rather than teachers. WEKA's widespread role in education has led to a repetitive

4. Currently in the development version only.

and distracting deluge of basic questions. Requests from students all over the world for assistance with their assignments and projects present a significant (and growing) problem; moreover, students often depart from proper mailing-list etiquette. It would have been better to identify the clientele for WEKA as a teaching tool, and offer a one-stop-shop for software, documentation and help that is distinct from the support infrastructure used by researchers.

All in all, WEKA has been a resounding success which we believe has significantly advanced the application of machine learning techniques in today's world. One of the most satisfying aspects of participating in the project is that the software has been incorporated into, and spawned, many other open-source projects.

Acknowledgments

We gratefully acknowledge the input of all contributors to the WEKA project, and want in particular to mention significant early contributions to the WEKA 3 codebase by Len Trigg and many later contributions by Richard Kirkby, Ashraf Kibriya and Xin Xu.

References

J. Bacon. *The Art of Community*. O'Reilly Media, 2009.

W. W. Cohen. Fast effective rule induction. In *Proc 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proc 15th International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, 1998.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

R. Kohavi, D. Sommerfield, and J. Dougherty. Data mining using MLC++ a machine learning library in C++. *International Journal on Artificial Intelligence Tools*, 6(4):537–566, 1997.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

Journal of Machine Learning Research 11 (2010) 2543-2596 Submitted 5/09; Revised 3/10; Published 10/10

Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization

Lin Xiao LIN.XIAO@MICROSOFT.COM
Microsoft Research
1 Microsoft Way
Redmond, WA 98052, USA

Editor: Sham Kakade

Abstract

We consider regularized stochastic learning and online optimization problems, where the objective function is the sum of two convex terms: one is the loss function of the learning task, and the other is a simple regularization term such as ℓ_1-norm for promoting sparsity. We develop extensions of Nesterov's dual averaging method, that can exploit the regularization structure in an online setting. At each iteration of these methods, the learning variables are adjusted by solving a simple minimization problem that involves the running average of all past subgradients of the loss function and the whole regularization term, not just its subgradient. In the case of ℓ_1-regularization, our method is particularly effective in obtaining sparse solutions. We show that these methods achieve the optimal convergence rates or regret bounds that are standard in the literature on stochastic and online convex optimization. For stochastic learning problems in which the loss functions have Lipschitz continuous gradients, we also present an accelerated version of the dual averaging method.

Keywords: stochastic learning, online optimization, ℓ_1-regularization, structural convex optimization, dual averaging methods, accelerated gradient methods

1. Introduction

In machine learning, online algorithms operate by repetitively drawing random examples, one at a time, and adjusting the learning variables using simple calculations that are usually based on the single example only. The low computational complexity (per iteration) of online algorithms is often associated with their slow convergence and low accuracy in solving the underlying optimization problems. As argued by Bottou and Bousquet (2008), the combined low complexity and low accuracy, together with other tradeoffs in statistical learning theory, still make online algorithms favorite choices for solving large-scale learning problems. Nevertheless, traditional online algorithms, such as stochastic gradient descent, have limited capability of exploiting problem structure in solving *regularized* learning problems. As a result, their low accuracy often makes it hard to obtain the desired regularization effects, for example, sparsity under ℓ_1-regularization.

In this paper, we develop a new class of online algorithms, the *regularized dual averaging* (RDA) methods, that can exploit the regularization structure more effectively in an online setting. In this section, we describe the two types of problems that we consider, and explain the motivation of our work.

1.1 Regularized Stochastic Learning

The regularized stochastic learning problems we consider are of the following form:

$$\underset{w}{\text{minimize}} \quad \Big\{ \phi(w) \triangleq \mathbf{E}_z f(w,z) + \Psi(w) \Big\} \tag{1}$$

where $w \in \mathbf{R}^n$ is the optimization variable (often called *weights* in learning problems), $z = (x,y)$ is an input-output pair of data drawn from an (unknown) underlying distribution, $f(w,z)$ is the loss function of using w and x to predict y, and $\Psi(w)$ is a regularization term. We assume $\Psi(w)$ is a closed convex function (Rockafellar, 1970, Section 7), and its effective domain, $\operatorname{dom}\Psi = \{w \in \mathbf{R}^n \,|\, \Psi(w) < +\infty\}$, is closed. We also assume that $f(w,z)$ is convex in w for each z, and it is subdifferentiable (a subgradient always exists) on $\operatorname{dom}\Psi$. Examples of the loss function $f(w,z)$ include:

- Least-squares: $x \in \mathbf{R}^n$, $y \in \mathbf{R}$, and $f(w,(x,y)) = (y - w^T x)^2$.
- Hinge loss: $x \in \mathbf{R}^n$, $y \in \{+1,-1\}$, and $f(w,(x,y)) = \max\{0, 1 - y(w^T x)\}$.
- Logistic regression: $x \in \mathbf{R}^n$, $y \in \{+1,-1\}$, and $f(w,(x,y)) = \log\big(1 + \exp\big(-y(w^T x)\big)\big)$.

Examples of the regularization term $\Psi(w)$ include:

- ℓ_1-regularization: $\Psi(w) = \lambda \|w\|_1$ with $\lambda > 0$. With ℓ_1-regularization, we hope to get a relatively sparse solution, that is, with many entries of the weight vector w being zeroes.
- ℓ_2-regularization: $\Psi(w) = (\sigma/2)\|w\|_2^2$, with $\sigma > 0$. When ℓ_2-regularization is used with the hinge loss function, we have the standard setup of support vector machines.
- Convex constraints: $\Psi(w)$ is the indicator function of a closed convex set C, that is,

$$\Psi(w) = I_C(w) \triangleq \begin{cases} 0, & \text{if } w \in C, \\ +\infty, & \text{otherwise.} \end{cases}$$

We can also consider mixed regularizations such as $\Psi(w) = \lambda\|w\|_1 + (\sigma/2)\|w\|_2^2$. These examples cover a wide range of practical problems in machine learning.

A common approach for solving stochastic learning problems is to approximate the expected loss function $\phi(w)$ by using a finite set of independent observations $z_1, \ldots, z_T$, and solve the following problem to minimize the empirical loss:

$$\underset{w}{\text{minimize}} \quad \frac{1}{T} \sum_{t=1}^{T} f(w, z_t) + \Psi(w). \tag{2}$$

By our assumptions, this is a convex optimization problem. Depending on the structure of particular problems, they can be solved efficiently by interior-point methods (e.g., Ferris and Munson, 2003; Koh et al., 2007), quasi-Newton methods (e.g., Andrew and Gao, 2007), or accelerated first-order methods (Nesterov, 2007; Tseng, 2008; Beck and Teboulle, 2009). However, this *batch optimization* approach may not scale well for very large problems: even with first-order methods, evaluating one single gradient of the objective function in (2) requires going through the whole data set.

In this paper, we consider *online algorithms* that process samples sequentially as they become available. More specifically, we draw a sequence of i.i.d. samples $z_1, z_2, z_3, \ldots$, and use them to

calculate a sequence $w_1, w_2, w_3, \ldots$. Suppose at time t, we have the most up-to-date weight vector w_t. Whenever z_t is available, we can evaluate the loss $f(w_t, z_t)$, and also a subgradient $g_t \in \partial f(w_t, z_t)$ (here $\partial f(w,z)$ denotes the subdifferential of $f(w,z)$ with respect to w). Then we compute w_{t+1} based on these information.

The most widely used online algorithm is the *stochastic gradient descent* (SGD) method. Consider the general case $\Psi(w) = I_C(w) + \psi(w)$, where $I_C(w)$ is a "hard" set constraint and $\psi(w)$ is a "soft" regularization. The SGD method takes the form

$$w_{t+1} = \Pi_C\big(w_t - \alpha_t\,(g_t + \xi_t)\big), \tag{3}$$

where α_t is an appropriate stepsize, ξ_t is a subgradient of ψ at w_t, and $\Pi_C(\cdot)$ denotes Euclidean projection onto the set $\mathcal{C}$. The SGD method belongs to the general scheme of *stochastic approximation*, which can be traced back to Robbins and Monro (1951) and Kiefer and Wolfowitz (1952). In general we are also allowed to use all previous information to compute w_{t+1}, and even second-order derivatives if the loss functions are smooth.

In a stochastic online setting, each weight vector w_t is a random variable that depends on $\{z_1, \ldots, z_{t-1}\}$, and so is the objective value $\phi(w_t)$. Assume an optimal solution $w^\star$ to the problem (1) exists, and let $\phi^\star = \phi(w^\star)$. The goal of online algorithms is to generate a sequence $\{w_t\}_{t=1}^\infty$ such that

$$\lim_{t\to\infty} \mathbf{E}\,\phi(w_t) = \phi^\star,$$

and hopefully with reasonable convergence rate. This is the case for the SGD method (3) if we choose the stepsize $\alpha_t = c/\sqrt{t}$, where c is a positive constant. The corresponding convergence rate is $O(1/\sqrt{t})$, which is indeed best possible for subgradient schemes with a *black-box* model, even in the case of deterministic optimization (Nemirovsky and Yudin, 1983). Despite such slow convergence and the associated low accuracy in the solutions (compared with batch optimization using, for example, interior-point methods), the SGD method has been very popular in the machine learning community due to its capability of scaling with very large data sets and good generalization performances observed in practice (e.g., Bottou and LeCun, 2004; Zhang, 2004; Shalev-Shwartz et al., 2007).

Nevertheless, a main drawback of the SGD method is its lack of capability in exploiting problem structure, especially for problems with explicit regularization. More specifically, the SGD method (3) treats the soft regularization $\psi(w)$ as a general convex function, and only uses its subgradient in computing the next weight vector. In this case, we can simply lump $\psi(w)$ into $f(w, z_t)$ and treat them as a single loss function. Although in theory the algorithm converges to an optimal solution (in expectation) as t goes to infinity, in practice it is usually stopped far before that. Even in the case of convergence in expectation, we still face (possibly big) variations in the solution due to the stochastic nature of the algorithm. Therefore, the regularization effect we hope to have by solving the problem (1) may be elusive for any particular solution generated by (3) based on finite random samples.

An important example and main motivation for this paper is ℓ_1-regularized stochastic learning, where $\Psi(w) = \lambda\|w\|_1$. In the case of batch learning, the empirical minimization problem (2) can be solved to very high precision, for example, by interior-point methods. Therefore simply rounding the weights with very small magnitudes toward zero is usually enough to produce desired sparsity. As a result, ℓ_1-regularization has been very effective in obtaining sparse solutions using the batch optimization approach in statistical learning (e.g., Tibshirani, 1996) and signal processing

(e.g., Chen et al., 1998). In contrast, the SGD method (3) hardly generates any sparse solution, and its inherent low accuracy makes the simple rounding approach very unreliable. Several principled soft-thresholding or truncation methods have been developed to address this problem (e.g., Langford et al., 2009; Duchi and Singer, 2009), but the levels of sparsity in their solutions are still unsatisfactory compared with the corresponding batch solutions.

In this paper, we develop *regularized dual averaging* (RDA) methods that can exploit the structure of (1) more effectively in a stochastic online setting. More specifically, each iteration of the RDA methods takes the form

$$w_{t+1} = \arg\min_{w} \left\{ \frac{1}{t} \sum_{\tau=1}^{t} \langle g_\tau, w \rangle + \Psi(w) + \frac{\beta_t}{t} h(w) \right\}, \tag{4}$$

where $h(w)$ is an auxiliary strongly convex function, and $\{\beta_t\}_{t\geq 1}$ is a nonnegative and nondecreasing input sequence, which determines the convergence properties of the algorithm. Essentially, at each iteration, this method minimizes the sum of three terms: a linear function obtained by averaging all previous subgradients (the dual average), the original regularization function $\Psi(w)$, and an additional strongly convex regularization term $(\beta_t/t)h(w)$. The RDA method is an extension of the *simple dual averaging* scheme of Nesterov (2009), which is equivalent to letting $\Psi(w)$ be the indicator function of a closed convex set.

For the RDA method to be practically efficient, we assume that the functions $\Psi(w)$ and $h(w)$ are *simple*, meaning that we are able to find a closed-form solution for the minimization problem in (4). Then the computational effort per iteration is only $O(n)$, the same as the SGD method. This assumption indeed holds in many cases. For example, if we let $\Psi(w) = \lambda\|w\|_1$ and $h(w) = (1/2)\|w\|_2^2$, then w_{t+1} has an entry-wise closed-from solution. This solution uses a much more aggressive truncation threshold than previous methods, thus results in significantly improved sparsity (see discussions in Section 5).

In terms of iteration complexity, we show that if $\beta_t = \Theta(\sqrt{t})$, that is, with order exactly $\sqrt{t}$, then the RDA method (4) has the standard convergence rate

$$\mathbf{E}\,\phi(\bar{w}_t) - \phi^\star \leq O\left(\frac{G}{\sqrt{t}}\right),$$

where $\bar{w}_t = (1/t)\sum_{\tau=1}^{t} w_\tau$ is the *primal average*, and G is a uniform upper bound on the norms of the subgradients g_t. If the regularization term $\Psi(w)$ is strongly convex, then setting $\beta_t \leq O(\ln t)$ gives a faster convergence rate $O(\ln t/t)$.

For stochastic optimization problems in which the loss functions $f(w,z)$ are all differentiable and have Lipschitz continuous gradients, we also develop an accelerated version of the RDA method that has the convergence rate

$$\mathbf{E}\,\phi(w_t) - \phi^\star \leq O(1)\left(\frac{L}{t^2} + \frac{Q}{\sqrt{t}}\right),$$

where L is the Lipschitz constant of the gradients, and Q^2 is an upper bound on the variances of the stochastic gradients. In addition to convergence in expectation, we show that the same orders of convergence rates hold with high probability.

1.2 Regularized Online Optimization

In *online optimization*, we use an online algorithm to generate a sequence of decisions w_t, one at a time, for $t = 1, 2, 3, \ldots$. At each time t, a previously unknown cost function f_t is revealed, and we encounter a loss $f_t(w_t)$. We assume that the cost functions f_t are convex for all $t \geq 1$. The goal of the online algorithm is to ensure that the total cost up to each time t, $\sum_{\tau=1}^{t} f_\tau(w_\tau)$, is not much larger than $\min_w \sum_{\tau=1}^{t} f_\tau(w)$, the smallest total cost of any fixed decision w from hindsight. The difference between these two cost is called the *regret* of the online algorithm. Applications of online optimization include online prediction of time series and sequential investment (e.g., Cesa-Bianchi and Lugosi, 2006).

In regularized online optimization, we add a convex regularization term $\Psi(w)$ to each cost function. The regret with respect to any fixed decision $w \in \operatorname{dom}\Psi$ is

$$R_t(w) \triangleq \sum_{\tau=1}^{t} \big(f_\tau(w_\tau) + \Psi(w_\tau)\big) - \sum_{\tau=1}^{t} \big(f_\tau(w) + \Psi(w)\big). \tag{5}$$

As in the stochastic setting, the online algorithm can query a subgradient $g_t \in \partial f_t(w_t)$ at each step, and possibly use all previous information, to compute the next decision w_{t+1}. It turns out that the simple subgradient method (3) is well suited for online optimization: with a stepsize $\alpha_t = \Theta(1/\sqrt{t})$, it has a regret $R_t(w) \leq O(\sqrt{t})$ for all $w \in \operatorname{dom}\Psi$ (Zinkevich, 2003). This regret bound cannot be improved in general for convex cost functions. However, if the cost functions are strongly convex, say with convexity parameter σ, then the same algorithm with stepsize $\alpha_t = 1/(\sigma t)$ gives an $O(\ln t)$ regret bound (e.g., Hazan et al., 2006; Bartlett et al., 2008).

Similar to the discussions on regularized stochastic learning, the online subgradient method (3) in general lacks the capability of exploiting the regularization structure. In this paper, we show that the same RDA method (4) can effectively exploit such structure in an online setting, and ensure the $O(\sqrt{t})$ regret bound with $\beta_t = \Theta(\sqrt{t})$. For strongly convex regularizations, setting $\beta_t = O(\ln t)$ yields the improved regret bound $O(\ln t)$.

Since there is no specifications on the probability distribution of the sequence of functions, nor assumptions like mutual independence, online optimization can be considered as a more general framework than stochastic learning. In this paper, we will first establish regret bounds of the RDA method for solving online optimization problems, then use them to derive convergence rates for solving stochastic learning problems.

1.3 Outline of Contents

The methods we develop apply to more general settings than $\mathbf{R}^n$ with Euclidean geometry. In Section 1.4, we introduce the necessary notations and definitions associated with a general finite-dimensional real vector space.

In Section 2, we present the generic RDA method for solving both the stochastic learning and online optimization problems, and give several concrete examples of the method.

In Section 3, we present the precise regret bounds of the RDA method for solving regularized online optimization problems.

In Section 4, we derive convergence rates of the RDA method for solving regularized stochastic learning problems. In addition to the rates of convergence in expectation, we also give associated high probability bounds.

In Section 5, we explain the connections of the RDA method to several related work, and analyze its capability of generating better sparse solutions than other methods.

In Section 6, we give an enhanced version of the ℓ_1-RDA method, and present computational experiments on the MNIST handwritten data set (LeCun et al., 1998). Our experiments show that the RDA method is capable of generate sparse solutions that are comparable to those obtained by batch learning using interior-point methods.

In Section 7, we discuss the RDA methods in the context of *structural convex optimization* and their connections to incremental subgradient methods. As an extension, we develop an accelerated version of the RDA method for stochastic optimization problems with smooth loss functions. We also discuss in detail the p-norm based RDA methods.

Appendices A-D contain technical proofs of our main results.

1.4 Notations and Generalities

Let $\mathcal{E}$ be a finite-dimensional real vector space, endowed with a norm $\|\cdot\|$. This norm defines a systems of balls: $\mathcal{B}(w,r) = \{u \in \mathcal{E} \,|\, \|u-w\| \le r\}$. Let $\mathcal{E}^*$ be the vector space of all linear functions on $\mathcal{E}$, and let $\langle s, w\rangle$ denote the value of $s \in \mathcal{E}^*$ at $w \in \mathcal{E}$. The dual space $\mathcal{E}^*$ is endowed with the dual norm $\|s\|_* = \max_{\|w\|\le 1} \langle s, w\rangle$.

A function $h : \mathcal{E} \to \mathbf{R} \cup \{+\infty\}$ is called *strongly convex* with respect to the norm $\|\cdot\|$ if there exists a constant $\sigma > 0$ such that

$$h(\alpha w + (1-\alpha)u) \le \alpha h(w) + (1-\alpha)h(u) - \frac{\sigma}{2}\alpha(1-\alpha)\|w-u\|^2, \qquad \forall\, w, u \in \operatorname{dom} h.$$

The constant σ is called the *convexity parameter*, or the *modulus* of strong convexity. Let $\operatorname{rint} \mathcal{C}$ denote the *relative interior* of a convex set $\mathcal{C}$ (Rockafellar, 1970, Section 6). If h is strongly convex with modulus σ, then for any $w \in \operatorname{dom} h$ and $u \in \operatorname{rint}(\operatorname{dom} h)$,

$$h(w) \ge h(u) + \langle s, w-u\rangle + \frac{\sigma}{2}\|w-u\|^2, \qquad \forall\, s \in \partial h(u).$$

See, for example, Goebel and Rockafellar (2008) and Juditsky and Nemirovski (2008).

In the special case of the coordinate vector space $\mathcal{E} = \mathbf{R}^n$, we have $\mathcal{E} = \mathcal{E}^*$, and the standard inner product $\langle s, w\rangle = s^T w = \sum_{i=1}^n s^{(i)} w^{(i)}$, where $w^{(i)}$ denotes the i-th coordinate of w. For the standard Euclidean norm, $\|w\| = \|w\|_2 = \sqrt{\langle w, w\rangle}$ and $\|s\|_* = \|s\|_2$. For any $w_0 \in \mathbf{R}^n$, the function $h(w) = (\sigma/2)\|w - w_0\|_2^2$ is strongly convex with modulus σ.

For another example, consider the ℓ_1-norm $\|w\| = \|w\|_1 = \sum_{i=1}^n |w^{(i)}|$ and its associated dual norm $\|w\|_* = \|w\|_\infty = \max_{1\le i\le n} |w^{(i)}|$. Let $\mathcal{S}_n$ be the standard simplex in $\mathbf{R}^n$, that is, $\mathcal{S}_n = \left\{w \in \mathbf{R}^n_+ \,|\, \sum_{i=1}^n w^{(i)} = 1\right\}$. Then the negative entropy function

$$h(w) = \sum_{i=1}^n w^{(i)} \ln w^{(i)} + \ln n, \tag{6}$$

with $\operatorname{dom} h = \mathcal{S}_n$, is strongly convex with respect to $\|\cdot\|_1$ with modulus 1 (see, e.g., Nesterov, 2005, Lemma 3). In this case, the unique minimizer of h is $w_0 = (1/n, \ldots, 1/n)$.

For a closed proper convex function Ψ, we use $\operatorname{Arg\,min}_w \Psi(w)$ to denote the (convex) set of minimizing solutions. If a convex function h has a unique minimizer, for example, when h is strongly convex, then we use $\arg\min_w h(w)$ to denote that single point.

Algorithm 1 Regularized dual averaging (RDA) method

input:

- an auxiliary function $h(w)$ that is strongly convex on $\operatorname{dom}\Psi$ and also satisfies

$$\arg\min_w h(w) \in \operatorname{Arg}\min_w \Psi(w). \tag{7}$$

- a nonnegative and nondecreasing sequence $\{\beta_t\}_{t\geq 1}$.

initialize: set $w_1 = \arg\min_w h(w)$ and $\bar{g}_0 = 0$.

for $t = 1, 2, 3, \ldots$ **do**

1. Given the function f_t, compute a subgradient $g_t \in \partial f_t(w_t)$.
2. Update the average subgradient:

$$\bar{g}_t = \frac{t-1}{t}\bar{g}_{t-1} + \frac{1}{t}g_t.$$

3. Compute the next weight vector:

$$w_{t+1} = \arg\min_w \left\{ \langle \bar{g}_t, w\rangle + \Psi(w) + \frac{\beta_t}{t}h(w) \right\}. \tag{8}$$

end for

2. Regularized Dual Averaging Method

In this section, we present the generic RDA method (Algorithm 1) for solving regularized stochastic learning and online optimization problems, and give several concrete examples. To unify notation, we use $f_t(w)$ to denote the cost function at each step t. For stochastic learning problems, we simply let $f_t(w) = f(w, z_t)$.

At the input to the RDA method, we need an auxiliary function h that is strongly convex on $\operatorname{dom}\Psi$. The condition (7) requires that its unique minimizer must also minimize the regularization function Ψ. This can be done, for example, by first choosing a starting point $w_0 \in \operatorname{Arg}\min_w \Psi(w)$ and an arbitrary strongly convex function $h'(w)$, then letting

$$h(w) = h'(w) - h'(w_0) - \langle \nabla h'(w_0), w - w_0\rangle.$$

In other words, $h(w)$ is the *Bregman divergence* from w_0 induced by $h'(w)$. If h' is not differentiable, but subdifferentiable at w_0, we can replace $\nabla h'(w_0)$ with a subgradient. The input sequence $\{\beta_t\}_{t\geq 1}$ determines the convergence rate, or regret bound, of the algorithm.

There are three steps in each iteration of the RDA method. Step 1 is to compute a subgradient of f_t at w_t, which is standard for all subgradient or gradient based methods. Step 2 is the online version of computing the average subgradient:

$$\bar{g}_t = \frac{1}{t}\sum_{\tau=1}^{t} g_\tau.$$

The name *dual averaging* comes from the fact that the subgradients live in the dual space $\mathcal{E}^*$.

Step 3 is most interesting and worth further explanation. In particular, the efficiency in computing w_{t+1} determines how useful the method is in practice. For this reason, we assume the regularization functions $\Psi(w)$ and $h(w)$ are *simple*. This means the minimization problem in (8) can be solved with little effort, especially if we are able to find a closed-form solution for w_{t+1}. At first sight, this assumption seems to be quite restrictive. However, the examples below show that this indeed is the case for many important learning problems in practice.

2.1 RDA Methods with General Convex Regularization

For a general convex regularization Ψ, we can choose any positive sequence $\{\beta_t\}_{t\geq 1}$ that is order exactly $\sqrt{t}$, to obtain an $O(1/\sqrt{t})$ convergence rate for stochastic learning, or an $O(\sqrt{t})$ regret bound for online optimization. We will state the formal convergence theorems in Sections 3 and 4. Here, we give several concrete examples. To be more specific, we choose a parameter $\gamma > 0$ and use the sequence

$$\beta_t = \gamma\sqrt{t}, \qquad t = 1, 2, 3, \ldots.$$

- *Nesterov's dual averaging method*. Let $\Psi(w)$ be the indicator function of a closed convex set $\mathcal{C}$. This recovers the *simple dual averaging* scheme in Nesterov (2009). If we choose $h(w) = (1/2)\|w\|_2^2$, then the Equation (8) yields

$$w_{t+1} = \Pi_{\mathcal{C}}\left(-\frac{\sqrt{t}}{\gamma}\bar{g}_t\right) = \Pi_{\mathcal{C}}\left(-\frac{1}{\gamma\sqrt{t}}\sum_{\tau=1}^{t} g_\tau\right). \tag{9}$$

  When $\mathcal{C} = \{w \in \mathbf{R}^n \,|\, \|w\|_1 \leq \delta\}$ for some $\delta > 0$, we have "hard" ℓ_1-regularization. In this case, although there is no closed-form solution for w_{t+1}, efficient algorithms for projection onto the ℓ_1-ball can be found, for example, in Duchi et al. (2008).

- *"Soft" ℓ_1-regularization*. Let $\Psi(w) = \lambda\|w\|_1$ for some $\lambda > 0$, and $h(w) = (1/2)\|w\|_2^2$. In this case, w_{t+1} has a closed-form solution (see Appendix A for the derivation):

$$w_{t+1}^{(i)} = \begin{cases} 0 & \text{if } \left|\bar{g}_t^{(i)}\right| \leq \lambda, \\ -\dfrac{\sqrt{t}}{\gamma}\left(\bar{g}_t^{(i)} - \lambda\,\mathrm{sgn}\left(\bar{g}_t^{(i)}\right)\right) & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, n. \tag{10}$$

  Here $\mathrm{sgn}(\cdot)$ is the *sign* or *signum* function, that is, $\mathrm{sgn}(\omega)$ equals 1 if $\omega > 0$, -1 if $\omega < 0$, and 0 if $\omega = 0$. Whenever a component of $\bar{g}_t$ is less than λ in magnitude, the corresponding component of w_{t+1} is set to zero. Further extensions of the ℓ_1-RDA method, and associated computational experiments, are given in Section 6.

- *Exponentiated dual averaging method*. Let $\Psi(w)$ be the indicator function of the standard simplex $\mathcal{S}_n$, and $h(w)$ be the negative entropy function defined in (6). In this case,

$$w_{t+1}^{(i)} = \frac{1}{Z_{t+1}}\exp\left(-\frac{\sqrt{t}}{\gamma}\bar{g}_t^{(i)}\right), \qquad i = 1, \ldots, n,$$

  where Z_{t+1} is a normalization parameter such that $\sum_{i=1}^n w_{t+1}^{(i)} = 1$. This is the dual averaging version of the exponentiated gradient algorithm (Kivinen and Warmuth, 1997); see also Tseng and Bertsekas (1993) and Juditsky et al. (2005). We note that this example is also covered by Nesterov's dual averaging method.

We discuss in detail the special case of p-norm RDA method in Section 7.2. Several other examples, including ℓ_∞-norm and a hybrid ℓ_1/ℓ_2-norm (*Berhu*) regularization, also admit closed-form solutions for w_{t+1}. Their solutions are similar in form to those obtained in the context of the FOBOS algorithm in Duchi and Singer (2009).

2.2 RDA Methods with Strongly Convex Regularization

If the regularization term $\Psi(w)$ is strongly convex, we can use any nonnegative and nondecreasing sequence $\{\beta_t\}_{t\geq 1}$ that grows no faster than $O(\ln t)$, to obtain an $O(\ln t/t)$ convergence rate for stochastic learning, or an $O(\ln t)$ regret bound for online optimization. For simplicity, in the following examples, we use the zero sequence $\beta_t = 0$ for all $t \geq 1$. In this case, we do not need the auxiliary function $h(w)$, and the Equation (8) becomes

$$w_{t+1} = \arg\min_w \left\{ \langle \bar{g}_t, w \rangle + \Psi(w) \right\}.$$

- *ℓ_2^2-regularization*. Let $\Psi(w) = (\sigma/2)\|w\|_2^2$ for some $\sigma > 0$. In this case,

$$w_{t+1} = -\frac{1}{\sigma}\bar{g}_t = -\frac{1}{\sigma t}\sum_{\tau=1}^{t} g_\tau.$$

- *Mixed ℓ_1/ℓ_2^2-regularization*. Let $\Psi(w) = \lambda\|w\|_1 + (\sigma/2)\|w\|_2^2$ with $\lambda > 0$ and $\sigma > 0$. In this case, we have

$$w_{t+1}^{(i)} = \begin{cases} 0 & \text{if } |\bar{g}_t^{(i)}| \leq \lambda, \\ -\frac{1}{\sigma}\left(\bar{g}_t^{(i)} - \lambda \operatorname{sgn}\left(\bar{g}_t^{(i)}\right)\right) & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, n.$$

- *Kullback-Leibler (KL) divergence regularization*. Let $\Psi(w) = \sigma D_{\mathrm{KL}}(w\|p)$, where the given probability distribution $p \in \operatorname{rint} \mathcal{S}_n$, and

$$D_{\mathrm{KL}}(w\|p) \triangleq \sum_{i=1}^{n} w^{(i)} \ln\left(\frac{w^{(i)}}{p^{(i)}}\right).$$

  Here $D_{\mathrm{KL}}(w\|p)$ is strongly convex with respect to $\|w\|_1$ with modulus 1. In this case,

$$w_{t+1}^{(i)} = \frac{1}{Z_{t+1}} p^{(i)} \exp\left(-\frac{1}{\sigma}\bar{g}_t^{(i)}\right),$$

  where Z_{t+1} is a normalization parameter such that $\sum_{i=1}^{n} w_{t+1}^{(i)} = 1$. KL divergence regularization has the *pseudo-sparsity* effect, meaning that most elements in w can be replaced by elements in the constant vector p without significantly increasing the loss function (e.g., Bradley and Bagnell, 2009).

3. Regret Bounds for Online Optimization

In this section, we give the precise regret bounds of the RDA method for solving regularized online optimization problems. The convergence rates for stochastic learning problems can be established based on these regret bounds, and will be given in the next section. For clarity, we gather here the general assumptions used throughout this paper:

- The regularization term $\Psi(w)$ is a closed proper convex function, and $\operatorname{dom}\Psi$ is closed. The symbol σ is dedicated to the convexity parameter of Ψ. Without loss of generality, we assume $\min_w \Psi(w) = 0$.
- For each $t \geq 1$, the function $f_t(w)$ is convex and subdifferentiable on $\operatorname{dom}\Psi$.
- The function $h(w)$ is strongly convex on $\operatorname{dom}\Psi$, and subdifferentiable on $\operatorname{rint}(\operatorname{dom}\Psi)$. Without loss of generality, assume $h(w)$ has convexity parameter 1 and $\min_w h(w) = 0$.

We will not repeat these general assumptions when stating our formal results later.

To facilitate regret analysis, we first give a few definitions. For any constant $D > 0$, we define the set

$$\mathcal{F}_D \triangleq \left\{ w \in \operatorname{dom}\Psi \;\middle|\; h(w) \leq D^2 \right\},$$

and let

$$\Gamma_D = \sup_{w \in \mathcal{F}_D} \inf_{g \in \partial\Psi(w)} \|g\|_*. \tag{11}$$

We use the convention $\inf_{g\in\emptyset} \|g\|_* = +\infty$, where $\emptyset$ denotes the empty set. As a result, if Ψ is not subdifferentiable everywhere on $\mathcal{F}_D$, that is, if $\partial\Psi(w) = \emptyset$ at some $w \in \mathcal{F}_D$, then we have $\Gamma_D = +\infty$. Note that Γ_D is not a Lipschitz-type constant which would be required to be an upper bound on all the subgradients; instead, we only require that at least one subgradient is bounded in norm by Γ_D at every point in the set $\mathcal{F}_D$.

We assume that the sequence of subgradients $\{g_t\}_{t\geq 1}$ generated by Algorithm 1 is bounded, that is, there exist a constant G such that

$$\|g_t\|_* \leq G, \quad \forall t \geq 1. \tag{12}$$

This is true, for example, if $\operatorname{dom}\Psi$ is compact and each f_t has Lipschitz-continuous gradient on $\operatorname{dom}\Psi$. We require that the input sequence $\{\beta_t\}_{t\geq 1}$ be chosen such that

$$\max\{\sigma, \beta_1\} > 0, \tag{13}$$

where σ is the convexity parameter of $\Psi(w)$. For convenience, we let $\beta_0 = \max\{\sigma, \beta_1\}$ and define the sequence of *regret bounds*

$$\Delta_t \triangleq \beta_t D^2 + \frac{G^2}{2} \sum_{\tau=0}^{t-1} \frac{1}{\sigma\tau + \beta_\tau} + \frac{2(\beta_0 - \beta_1)G^2}{(\beta_1 + \sigma)^2}, \qquad t = 1, 2, 3, \ldots, \tag{14}$$

where D is the constant used in the definition of $\mathcal{F}_D$. We could always set $\beta_1 \geq \sigma$, so that $\beta_0 = \beta_1$ and therefore the term $2(\beta_0 - \beta_1)G^2/(\beta_1 + \sigma)^2$ vanishes in the definition (14). However, when $\sigma > 0$, we would like to keep the flexibility of setting $\beta_t = 0$ for all $t \geq 1$, as we did in Section 2.2.

Theorem 1 *Let the sequences $\{w_t\}_{t\geq 1}$ and $\{g_t\}_{t\geq 1}$ be generated by Algorithm 1, and assume (12) and (13) hold. Then for any $t \geq 1$ and any $w \in \mathcal{F}_D$, we have:*

(a) The regret defined in (5) is bounded by Δ_t, that is,

$$R_t(w) \leq \Delta_t. \tag{15}$$

(b) The primal variables are bounded as

$$\|w_{t+1} - w\|^2 \leq \frac{2}{\sigma t + \beta_t}\big(\Delta_t - R_t(w)\big). \tag{16}$$

(c) If w is an interior point, that is, $\mathcal{B}(w,r) \subset \mathcal{F}_D$ for some $r > 0$, then

$$\|\bar{g}_t\|_* \leq \Gamma_D - \frac{1}{2}\sigma r + \frac{1}{rt}\big(\Delta_t - R_t(w)\big). \tag{17}$$

In Theorem 1, the bounds on $\|w_{t+1} - w\|^2$ and $\|\bar{g}_t\|_*$ depend on the regret $R_t(w)$. More precisely, they depend on $\Delta_t - R_t(w)$, which is the *slack* of the regret bound in (15). A smaller slack is equivalent to a larger regret $R_t(w)$, which means w is a better *fixed* solution for the online optimization problem (the best one gives the largest regret); correspondingly, the inequality (16) gives a tighter bound on $\|w_{t+1} - w\|^2$. In (17), the left-hand side $\|\bar{g}_t\|_*$ does not depend on any particular interior point w to compare with, but the right-hand side depends on both $R_t(w)$ and how far w is from the boundary of $\mathcal{F}_D$. The tightest bound on $\|\bar{g}_t\|_*$ can be obtained by taking the infimum of the right-hand side over all $w \in \operatorname{int} \mathcal{F}_D$. We further elaborate on part (c) through the following two examples:

- Consider the case when Ψ is the indicator function of a closed convex set $\mathcal{C}$. In this case, $\sigma = 0$ and $\partial\Psi(w)$ is the *normal cone* to $\mathcal{C}$ at w (Rockafellar, 1970, Section 23). By the definition (11), we have $\Gamma_D = 0$ because the zero vector is a subgradient at every $w \in \mathcal{C}$, even though the normal cones can be unbounded at the boundary of $\mathcal{C}$. In this case, if $\mathcal{B}(w,r) \subset \mathcal{F}_D$ for some $r > 0$, then (17) simplifies to

$$\|\bar{g}_t\|_* \leq \frac{1}{rt}\big(\Delta_t - R_t(w)\big).$$

- Consider the function $\Psi(w) = \sigma D_{\mathrm{KL}}(w\|p)$ with $\operatorname{dom}\Psi = \mathcal{S}_n$ (assuming $p \in \operatorname{rint}\mathcal{S}_n$). In this case, $\operatorname{dom}\Psi$, and hence $\mathcal{F}_D$, have empty interior. Therefore the bound in part (c) does not apply. In fact, the quantity Γ_D can be unbounded anyway. In particular, the subdifferentials of Ψ at the relative boundary of $\mathcal{S}_n$ are all empty. In the relative interior of $\mathcal{S}_n$, the subgradients (actually gradients) of Ψ always exist, but can become unbounded for points approaching the relative boundary. Nevertheless, the bounds in parts (a) and (b) still hold.

The proof of Theorem 1 is given in Appendix B. In the rest of this section, we discuss more concrete regret bounds depending on whether or not Ψ is strongly convex.

3.1 Regret Bound with General Convex Regularization

For a general convex regularization term Ψ, any nonnegative and nondecreasing sequence $\beta_t = \Theta(\sqrt{t})$ gives an $O(\sqrt{t})$ regret bound. Here we give detailed analysis for the sequence used in Section 2.1. More specifically, we choose a constant $\gamma > 0$ and let

$$\beta_t = \gamma\sqrt{t}, \qquad \forall\, t \geq 1. \tag{18}$$

We have the following corollary of Theorem 1.

Corollary 2 *Let the sequences $\{w_t\}_{t\geq 1}$ and $\{g_t\}_{t\geq 1}$ be generated by Algorithm 1 using $\{\beta_t\}_{t\geq 1}$ defined in (18), and assume (12) holds. Then for any $t \geq 1$ and any $w \in \mathcal{F}_D$:*

(a) The regret is bounded as

$$R_t(w) \leq \left(\gamma D^2 + \frac{G^2}{\gamma}\right)\sqrt{t}.$$

(b) The primal variables are bounded as

$$\frac{1}{2}\|w_{t+1} - w\|^2 \leq D^2 + \frac{G^2}{\gamma^2} - \frac{1}{\gamma\sqrt{t}}R_t(w).$$

(c) If w is an interior point, that is, $\mathcal{B}(w,r) \subset \mathcal{F}_D$ for some $r > 0$, then

$$\|\bar{g}_t\|_* \leq \Gamma_D + \left(\gamma D^2 + \frac{G^2}{\gamma}\right)\frac{1}{r\sqrt{t}} - \frac{1}{rt}R_t(w).$$

Proof To simplify regret analysis, let $\gamma \geq \sigma$. Therefore $\beta_0 = \beta_1 = \gamma$. Then Δ_t defined in (14) becomes

$$\Delta_t = \gamma\sqrt{t}D^2 + \frac{G^2}{2\gamma}\left(1 + \sum_{\tau=1}^{t-1}\frac{1}{\sqrt{\tau}}\right).$$

Next using the inequality

$$\sum_{\tau=1}^{t-1}\frac{1}{\sqrt{\tau}} \leq 1 + \int_1^t \frac{1}{\sqrt{\tau}}d\tau = 2\sqrt{t} - 1,$$

we get

$$\Delta_t \leq \gamma\sqrt{t}D^2 + \frac{G^2}{2\gamma}\left(1 + \left(2\sqrt{t} - 1\right)\right) = \left(\gamma D^2 + \frac{G^2}{\gamma}\right)\sqrt{t}.$$

Combining the above inequality and the conclusions of Theorem 1 proves the corollary. ■

The regret bound in Corollary 2 is essentially the same as the *online gradient descent* method of Zinkevich (2003), which has the form (3), with the stepsize $\alpha_t = 1/(\gamma\sqrt{t})$. The main advantage of the RDA method is its capability of exploiting the regularization structure, as shown in Section 2. The parameters D and G are not used explicitly in the algorithm. However, we need good estimates of them for choosing a reasonable value for γ. The best γ that minimizes the expression $\gamma D^2 + G^2/\gamma$ is

$$\gamma^\star = \frac{G}{D},$$

which leads to the simplified regret bound

$$R_t(w) \leq 2GD\sqrt{t}.$$

If the total number of online iterations T is known in advance, then using a constant stepsize in the classical gradient method (3), say

$$\alpha_t = \frac{1}{\gamma^\star}\sqrt{\frac{2}{T}} = \frac{D}{G}\sqrt{\frac{2}{T}}, \qquad \forall t = 1,\ldots,T, \tag{19}$$

gives a slightly improved bound $R_T(w) \leq \sqrt{2}GD\sqrt{T}$ (see, e.g., Nemirovski et al., 2009).

The bound in part (b) does not converge to zero. This result is still interesting because there is no special caution taken in the RDA method, more specifically in (8), to ensure the boundedness of the sequence w_t. In the case $\Psi(w) = 0$, as pointed out by Nesterov (2009), this may even look surprising since we are minimizing over $\mathcal{E}$ the sum of a linear function and a regularization term $(\gamma/\sqrt{t})h(w)$ that eventually goes to zero.

Part (c) gives a bound on the norm of the dual average. If $\Psi(w)$ is the indicator function of a closed convex set, then $\Gamma_D = 0$ and part (c) shows that $\bar{g}_t$ actually converges to zero if there exist an interior w in $\mathcal{F}_D$ such that $R_t(w) \geq 0$. However, a properly scaled version of $\bar{g}_t$, $-(\sqrt{t}/\gamma)\bar{g}_t$, tracks the optimal solution; see the examples in Section 2.1.

3.2 Regret Bounds with Strongly Convex Regularization

If the regularization function $\Psi(w)$ is strongly convex, that is, with a convexity parameter $\sigma > 0$, then any nonnegative, nondecreasing sequence that satisfies $\beta_t \leq O(\ln t)$ will give an $O(\ln t)$ regret bound. If $\{\beta_t\}_{t\geq 1}$ is not the all zero sequence, we can simply choose the auxiliary function $h(w) = (1/\sigma)\Psi(w)$. Here are several possibilities:

- *Positive constant sequences*. For simplicity, let $\beta_t = \sigma$ for $t \geq 0$. In this case,

$$\Delta_t = \sigma D^2 + \frac{G^2}{2\sigma}\sum_{\tau=0}^{t-1}\frac{1}{\tau+1} \leq \sigma D^2 + \frac{G^2}{2\sigma}(1+\ln t).$$

- *Logarithmic sequences*. Let $\beta_t = \sigma(1+\ln t)$ for $t \geq 1$. In this case, $\beta_0 = \beta_1 = \sigma$ and

$$\Delta_t = \sigma(1+\ln t)D^2 + \frac{G^2}{2\sigma}\left(1 + \sum_{\tau=1}^{t-1}\frac{1}{\tau+1+\ln\tau}\right) \leq \left(\sigma D^2 + \frac{G^2}{2\sigma}\right)(1+\ln t).$$

- *The zero sequence*. Let $\beta_t = 0$ for $t \geq 1$. In this case, $\beta_0 = \sigma$ and

$$\Delta_t = \frac{G^2}{2\sigma}\left(1 + \sum_{\tau=1}^{t-1}\frac{1}{\tau}\right) + \frac{2G^2}{\sigma} \leq \frac{G^2}{2\sigma}(6+\ln t). \tag{20}$$

Notice that in this last case, the regret bound does not depend on D.

When Ψ is strongly convex, we also conclude that, given two different points u and v, the regrets $R_t(u)$ and $R_t(v)$ cannot be nonnegative simultaneously if t is large enough. To see this, we notice that if $R_t(u)$ and $R_t(v)$ are nonnegative simultaneously for some t, then part (b) of Theorem 1 implies

$$\|w_{t+1} - u\|^2 \leq O\left(\frac{\ln t}{t}\right), \qquad \text{and} \qquad \|w_{t+1} - v\|^2 \leq O\left(\frac{\ln t}{t}\right),$$

which again implies

$$\|u - v\|^2 \leq (\|w_{t+1} - u\| + \|w_{t+1} - v\|)^2 \leq O\left(\frac{\ln t}{t}\right).$$

Therefore, if the event $R_t(u) \geq 0$ and $R_t(v) \geq 0$ happens for infinitely many t, we must have $u = v$. If $u \neq v$, then eventually at least one of the regrets associated with them will become negative. However, it is possible to construct sequences of functions f_t such that the points with nonnegative regrets do not converge to a fixed point.

4. Convergence Rates for Stochastic Learning

In this section, we give convergence rates of the RDA method when it is used to solve the regularized stochastic learning problem (1), and also the related high probability bounds. These rates and bounds are established not for the individual w_t's generated by the RDA method, but rather for the *primal average*

$$\bar{w}_t = \frac{1}{t}\sum_{\tau=1}^{t} w_\tau, \qquad t \geq 1.$$

4.1 Rate of Convergence in Expectation

Theorem 3 *Assume there exists an optimal solution $w^\star$ to the problem (1) that satisfies $h(w^\star) \leq D^2$ for some $D > 0$, and let $\phi^\star = \phi(w^\star)$. Let the sequences $\{w_t\}_{t\geq 1}$ and $\{g_t\}_{t\geq 1}$ be generated by Algorithm 1, and assume (12) holds. Then for any $t \geq 1$, we have:*

(a) The expected cost associated with the random variable $\bar{w}_t$ is bounded as

$$\mathbf{E}\,\phi(\bar{w}_t) - \phi^\star \leq \frac{1}{t}\Delta_t.$$

(b) The primal variables are bounded as

$$\mathbf{E}\,\|w_{t+1} - w^\star\|^2 \leq \frac{2}{\sigma t + \beta_t}\Delta_t.$$

(c) If $w^\star$ is an interior point, that is, $\mathcal{B}(w^\star, r) \subset \mathcal{F}_D$ for some $r > 0$, then

$$\mathbf{E}\|\bar{g}_t\|_* \leq \Gamma_D - \frac{1}{2}\sigma r + \frac{1}{rt}\Delta_t.$$

Proof First, we substitute all $f_\tau(\cdot)$ by $f(\cdot, z_\tau)$ in the definition of the regret

$$R_t(w^\star) = \sum_{\tau=1}^{t} \big(f(w_\tau, z_\tau) + \Psi(w_\tau)\big) - \sum_{\tau=1}^{t} \big(f(w^\star, z_\tau) + \Psi(w^\star)\big).$$

Let $\mathbf{z}[t]$ denote the collection of i.i.d. random variables $(z_1, \ldots, z_t)$. All the expectations in Theorem 3 are taken with respect to $\mathbf{z}[t]$, that is, the symbol $\mathbf{E}$ can be written more explicitly as $\mathbf{E}_{\mathbf{z}[t]}$. We note that the random variable w_τ, where $1 \le \tau \le t$, is a function of $(z_1, \ldots, z_{\tau-1})$, and is independent of $(z_\tau, \ldots, z_t)$. Therefore

$$\mathbf{E}_{\mathbf{z}[t]}\big(f(w_\tau, z_\tau) + \Psi(w_\tau)\big) = \mathbf{E}_{\mathbf{z}[\tau-1]}\big(\mathbf{E}_{z_\tau} f(w_\tau, z_\tau) + \Psi(w_\tau)\big) = \mathbf{E}_{\mathbf{z}[\tau-1]}\phi(w_\tau) = \mathbf{E}_{\mathbf{z}[t]}\phi(w_\tau),$$

and

$$\mathbf{E}_{\mathbf{z}[t]}\big(f(w^\star, z_\tau) + \Psi(w^\star)\big) = \mathbf{E}_{z_\tau} f(w^\star, z_\tau) + \Psi(w^\star) = \phi(w^\star) = \phi^\star.$$

Since $\phi^\star = \phi(w^\star) = \min_w \phi(w)$, we have

$$\mathbf{E}_{\mathbf{z}[t]} R_t(w^\star) = \sum_{\tau=1}^{t} \mathbf{E}_{\mathbf{z}[t]}\phi(w_\tau) - t\phi^\star \ge 0. \tag{21}$$

By convexity of ϕ, we have

$$\phi(\bar{w}_t) = \phi\left(\frac{1}{t}\sum_{\tau=1}^{t} w_\tau\right) \le \frac{1}{t}\sum_{\tau=1}^{t} \phi(w_\tau)$$

Taking expectation with respect to $\mathbf{z}[t]$ and subtracting $\phi^\star$, we have

$$\mathbf{E}_{\mathbf{z}[t]}\phi(\bar{w}_t) - \phi^\star \le \frac{1}{t}\left(\sum_{\tau=1}^{t} \mathbf{E}_{\mathbf{z}[t]}\phi(\bar{w}_\tau) - t\phi^\star\right) = \frac{1}{t}\mathbf{E}_{\mathbf{z}[t]} R_t(w^\star).$$

Then part (a) follows from that of Theorem 1, which states that $R_t(w^\star) \le \Delta_t$ for all realizations of $\mathbf{z}[t]$. Similarly, parts (b) and (c) follow from those of Theorem 1 and (21). ■

Specific convergence rates can be obtained in parallel with the regret bounds discussed in Sections 3.1 and 3.2. We only need to divide every regret bound by t to obtain the corresponding rate of convergence in expectation. More specifically, using appropriate sequences $\{\beta_t\}_{t\ge1}$, we have $\mathbf{E}\phi(\bar{w}_t)$ converging to $\phi^\star$ with rate $O(1/\sqrt{t})$ for general convex regularization, and $O(\ln t/t)$ for strongly convex regularization.

The bound in part (b) applies to both the case $\sigma = 0$ and the case $\sigma > 0$. For the latter, we can derive a slightly different and more specific bound. When Ψ has convexity parameter $\sigma > 0$, so is the function ϕ. Therefore,

$$\phi(w_t) \ge \phi(w^\star) + \langle s, w_t - w^\star\rangle + \frac{\sigma}{2}\|w_t - w^\star\|^2, \qquad \forall s \in \partial\phi(w^\star).$$

Since $w^\star$ is the minimizer of ϕ, we must have $0 \in \partial\phi(w^\star)$ (Rockafellar, 1970, Section 27). Setting $s = 0$ in the above inequality and rearranging terms, we have

$$\|w_t - w^\star\|^2 \le \frac{2}{\sigma}\left(\phi(w_t) - \phi^\star\right).$$

Taking expectation of both sides of the above inequality leads to

$$\mathbf{E}\|w_t - w^\star\|^2 \leq \frac{2}{\sigma}\left(\mathbf{E}\phi(w_t) - \phi^\star\right) \leq \frac{2}{\sigma t}\Delta_t, \tag{22}$$

where in the last step we used part (a) of Theorem 3. This bound directly relate w_t to Δ_t.

Next we take a closer look at the quantity $\mathbf{E}\|\bar{w}_t - w^\star\|^2$. By convexity of $\|\cdot\|^2$, we have

$$\mathbf{E}\|\bar{w}_t - w^\star\|^2 \leq \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{E}\|w_\tau - w^\star\|^2 \tag{23}$$

If $\sigma = 0$, then it is simply bounded by a constant because each $\mathbf{E}\|w_\tau - w^\star\|^2$ for $1 \leq \tau \leq t$ is bounded by a constant. When $\sigma > 0$, the optimal solution $w^\star$ is unique, and we have:

Corollary 4 *If Ψ is strongly convex with convexity parameter $\sigma > 0$ and $\beta_t = O(\ln t)$, then*

$$\mathbf{E}\|\bar{w}_t - w^\star\|^2 \leq O\left(\frac{(\ln t)^2}{t}\right).$$

Proof For the ease of presentation, we consider the case $\beta_t = 0$ for all $t \geq 1$. Substituting the bound on Δ_t in (20) into the inequality (22) gives

$$\mathbf{E}\|w_t - w^\star\|^2 \leq \frac{(6 + \ln t)\,G^2}{t\,\sigma^2}, \qquad \forall t \geq 1.$$

Then by (23),

$$\mathbf{E}\|\bar{w}_t - w^\star\|^2 \leq \frac{1}{t}\sum_{\tau=1}^{t}\left(\frac{6}{\tau} + \frac{\ln\tau}{\tau}\right)\frac{G^2}{\sigma^2} \leq \frac{1}{t}\left(6(1+\ln t) + \frac{1}{2}(\ln t)^2\right)\frac{G^2}{\sigma^2}.$$

In other words, $\mathbf{E}\|\bar{w}_t - w^\star\|^2$ converges to zero with rate $O((\ln t)^2/t)$. This can be shown for any $\beta_t = O(\ln t)$; see Section 3.2 for other choices of β_t. ■

As a further note, the conclusions in Theorem 3 still hold if the assumption (12) is weakened to

$$\mathbf{E}\,\|g_t\|_*^2 \leq G^2, \qquad \forall t \geq 1. \tag{24}$$

However, we need (12) in order to prove the high probability bounds presented next.

4.2 High Probability Bounds

For stochastic learning problems, in addition to the rates of convergence in expectation, it is often desirable to obtain confidence level bounds for approximate solutions. For this purpose, we start from part (a) of Theorem 3, which states $\mathbf{E}\phi(w_t) - \phi^\star \leq (1/t)\Delta_t$. By Markov's inequality, we have for any $\varepsilon > 0$,

$$\mathrm{Prob}\big(\phi(\bar{w}_t) - \phi^\star > \varepsilon\big) \leq \frac{\Delta_t}{\varepsilon t}. \tag{25}$$

This bound holds even with the weakened assumption (24). However, it is possible to have much tighter bounds under more restrictive assumptions. To this end, we have the following result.

Theorem 5 *Assume there exist constants D and G such that $h(w^\star) \le D^2$, and $h(w_t) \le D^2$ and $\|g_t\|_* \le G$ for all $t \ge 1$. Then for any $\delta \in (0,1)$, we have, with probability at least $1-\delta$,*

$$\phi(\bar{w}_t) - \phi^\star \le \frac{\Delta_t}{t} + \frac{8GD\sqrt{\ln(1/\delta)}}{\sqrt{t}}, \qquad \forall t \ge 1. \tag{26}$$

Theorem 5 is proved in Appendix C.

From our results in Section 3.1, with the input sequence $\beta_t = \gamma\sqrt{t}$ for all $t \ge 1$, we have $\Delta_t = O(\sqrt{t})$ regardless of $\sigma = 0$ or $\sigma > 0$. Therefore, $\phi(\bar{w}_t) - \phi^\star = O(1/\sqrt{t})$ with high probability. To simplify further discussion, let $\gamma = G/D$, hence $\Delta_t \le 2GD\sqrt{t}$ (see Section 3.1). In this case, if $\delta \le 1/e \approx 0.368$, then with probability at least $1-\delta$,

$$\phi(\bar{w}_t) - \phi^\star \le \frac{10GD\sqrt{\ln(1/\delta)}}{\sqrt{t}}.$$

Letting $\varepsilon = 10GD\sqrt{\ln(1/\delta)}/\sqrt{t}$, then the above bound is equivalent to

$$\text{Prob}\left(\phi(\bar{w}_t) - \phi^\star > \varepsilon\right) \le \exp\left(-\frac{\varepsilon^2 t}{(10GD)^2}\right),$$

which is much tighter than the one in (25). It follows that for any chosen accuracy ε and $0 < \delta \le 1/e$, the sample size

$$t \ge \frac{(10GD)^2 \ln(1/\delta)}{\varepsilon^2}$$

guarantees that, with probability at least $1-\delta$, $\bar{w}_t$ is an ε-optimal solution of the original stochastic optimization problem (1).

When Ψ is strongly convex ($\sigma > 0$), our results in Section 3.2 show that we can obtain regret bounds $\Delta_t = O(\ln t)$ using $\beta_t = O(\ln t)$. However, the high probability bound in Theorem 5 does not improve: we still have $\phi(\bar{w}_t) - \phi^\star = O(1/\sqrt{t})$, not $O(\ln t/t)$. The reason is that the concentration inequality (Azuma, 1967) used in proving Theorem 5 cannot take advantage of the strong-convexity property. By using a refined concentration inequality due to Freedman (1975), Kakade and Tewari (2009, Theorem 2) showed that for strongly convex stochastic learning problems, with probability at least $1 - 4\delta \ln t$,

$$\phi(\bar{w}_t) - \phi^\star \le \frac{R_t(w^\star)}{t} + 4\frac{\sqrt{R_t(w^\star)}}{t}\sqrt{\frac{G^2\ln(1/\delta)}{\sigma}} + \max\left\{\frac{16G^2}{\sigma}, 6B\right\}\frac{\ln(1/\delta)}{t}.$$

In our context, the constant B is an upper bound on $f(w,z) + \Phi(w)$ for $w \in \mathcal{F}_D$. Using the regret bound $R(w^\star) \le \Delta_t$, this gives

$$\phi(\bar{w}_t) - \phi^\star \le \frac{\Delta_t}{t} + O\left(\frac{\sqrt{\Delta_t \ln(1/\delta)}}{t} + \frac{\ln(1/\delta)}{t}\right).$$

Here the constants hidden in the O-notation are determined by G, σ and D. Plugging in $\Delta_t = O(\ln t)$, we have $\phi(\bar{w}_t) - \phi^\star = O(\ln t/t)$ with high probability. The additional penalty of getting the high probability bound, compared with the rate of convergence in expectation, is only $O(\sqrt{\ln t}/t)$.

5. Related Work

As we pointed out in Section 2.1, if Ψ is the indicator function of a convex set $\mathcal{C}$, then the RDA method recovers the simple dual averaging scheme in Nesterov (2009). This special case also belongs to a more general primal-dual algorithmic framework developed by Shalev-Shwartz and Singer (2006), which can be expressed equivalently in our notation:

$$w_{t+1} = \arg\min_{w \in \mathcal{C}} \left\{ \frac{1}{\gamma\sqrt{t}} \left\langle \sum_{\tau=1}^{t} d_\tau^t, w \right\rangle + h(w) \right\},$$

where $(d_1^t, \ldots, d_t^t)$ is the set of dual variables that can be chosen at time t. The simple dual averaging scheme (9) is in fact the *passive* extreme of their framework in which the dual variables are simply chosen as the subgradients and do not change over time, that is,

$$d_\tau^t = g_\tau, \qquad \forall \tau \le t, \qquad \forall t \ge 1. \tag{27}$$

However, with the addition of a general regularization term $\Psi(w)$ as in (4), the convergence analysis and $O(\sqrt{t})$ regret bound of the RDA method do *not* follow directly as corollaries of either Nesterov (2009) or Shalev-Shwartz and Singer (2006). Our analysis in Appendix B extends the framework of Nesterov (2009).

Shalev-Shwartz and Kakade (2009) extended the primal-dual framework of Shalev-Shwartz and Singer (2006) to strongly convex functions and obtained $O(\ln t)$ regret bound. In the context of this paper, their algorithm takes the form

$$w_{t+1} = \arg\min_{w \in \mathcal{C}} \left\{ \frac{1}{\sigma t} \left\langle \sum_{\tau=1}^{t} d_\tau^t, w \right\rangle + h(w) \right\},$$

where σ is the convexity parameter of Ψ, and $h(w) = (1/\sigma)\Psi(w)$. The passive extreme of this method, with the dual variables chosen in (27), is equivalent to a special case of the RDA method with $\beta_t = 0$ for all $t \ge 1$.

Other than improving the iteration complexity, the idea of treating the regularization explicitly in each step of a subgradient-based method (instead of lumping it together with the loss function and taking their subgradients) is mainly motivated by practical considerations, such as obtaining sparse solutions. In the case of ℓ_1-regularization, this leads to soft-thresholding type of algorithms, in both batch learning (e.g., Figueiredo et al., 2007; Wright et al., 2009; Bredies and Lorenz, 2008; Beck and Teboulle, 2009) and the online setting (e.g., Langford et al., 2009; Duchi and Singer, 2009; Shalev-Shwartz and Tewari, 2009). Most of these algorithms can be viewed as extensions of classical gradient methods (including mirror-descent methods) in which the new iterate is obtained by stepping from the current iterate along a single subgradient, and then followed by a truncation. Other types of algorithms include an interior-point based stochastic approximation scheme by Carbonetto et al. (2009), and Balakrishnan and Madigan (2008), where a modified shrinkage algorithm is developed based on sequential quadratic approximations of the loss function.

The main point of this paper, is to show that dual-averaging based methods can be more effective in exploiting the regularization structure, especially in a stochastic or online setting. To demonstrate this point, we compare the RDA method with the FOBOS method studied in Duchi and Singer (2009). In an online setting, each iteration of the FOBOS method consists of the following two

steps:

$$w_{t+\frac{1}{2}} = w_t - \alpha_t g_t,$$
$$w_{t+1} = \arg\min_w \left\{ \frac{1}{2} \left\| w - w_{t+\frac{1}{2}} \right\|_2^2 + \alpha_t \Psi(w) \right\}.$$

For convergence with optimal rates, the stepsize α_t is set to be $\Theta(1/\sqrt{t})$ for general convex regularizations and $\Theta(1/t)$ if Ψ is strongly convex. This method is based on a technique known as *forward-backward splitting*, which was first proposed by Lions and Mercier (1979) and later analyzed by Chen and Rockafellar (1997) and Tseng (2000). For easy comparison with the RDA method, we rewrite the FOBOS method in an equivalent form

$$w_{t+1} = \arg\min_w \left\{ \langle g_t, w \rangle + \Psi(w) + \frac{1}{2\alpha_t} \|w - w_t\|_2^2 \right\}. \tag{28}$$

Compared with this form of the FOBOS method, the RDA method (8) uses the average subgradient $\bar{g}_t$ instead of the current subgradient g_t; it uses a global proximal function, say $h(w) = (1/2)\|w\|_2^2$, instead of its local Bregman divergence $(1/2)\|w - w_t\|_2^2$; moreover, the coefficient for the proximal function is $\beta_t/t = \Theta(1/\sqrt{t})$ instead of $1/\alpha_t = \Theta(\sqrt{t})$ for general convex regularization, and $O(\ln t/t)$ instead of $\Theta(t)$ for strongly convex regularization. Although these two methods have the same order of iteration complexity, the differences list above contribute to quite different properties of their solutions.

These differences can be better understood in the special case of ℓ_1-regularization, that is, when $\Psi(w) = \lambda\|w\|_1$. In this case, the FOBOS method is equivalent to a special case of the *Truncated Gradient* (TG) method of Langford et al. (2009). The TG method truncates the solutions obtained by the standard SGD method every K steps; more specifically,

$$w_{t+1}^{(i)} = \begin{cases} \text{trnc}\left(w_t^{(i)} - \alpha_t g_t^{(i)}, \lambda_t^{\text{TG}}, \theta \right) & \text{if } \text{mod}(t,K) = 0, \\ w_t^{(i)} - \alpha_t g_t^{(i)} & \text{otherwise}, \end{cases} \tag{29}$$

where $\lambda_t^{\text{TG}} = \alpha_t \lambda K$, $\text{mod}(t,K)$ is the remainder on division of t by K, and

$$\text{trnc}(\omega, \lambda_t^{\text{TG}}, \theta) = \begin{cases} 0 & \text{if } |\omega| \le \lambda_t^{\text{TG}}, \\ \omega - \lambda_t^{\text{TG}} \,\text{sgn}(\omega) & \text{if } \lambda_t^{\text{TG}} < |\omega| \le \theta, \\ \omega & \text{if } |\omega| > \theta. \end{cases}$$

When $K = 1$ and $\theta = +\infty$, the TG method is the same as the FOBOS method (28) with ℓ_1-regularization. Now comparing the truncation threshold λ_t^{TG} and the threshold λ used in the ℓ_1-RDA method (10): with $\alpha_t = \Theta(1/\sqrt{t})$, we have $\lambda_t^{\text{TG}} = \Theta(1/\sqrt{t})\lambda$. This $\Theta(1/\sqrt{t})$ discount factor is also common for other previous work that use soft-thresholding, including Shalev-Shwartz and Tewari (2009). It is clear that the RDA method uses a much more aggressive truncation threshold, thus is able to generate significantly more sparse solutions. This is confirmed by our computational experiments in the next section.

Most recently, Duchi et al. (2010) developed a family of subgradient methods that can adaptively modifying the proximal function (squared Mahalanobis norms) at each iteration, in order to better incorporate learned knowledge about geometry of the data. Their methods include extensions for both the mirror-descent type of algorithms like (28) and the RDA methods studied in this paper.

Algorithm 2 Enhanced ℓ_1-RDA method

Input: $\gamma > 0, \rho \geq 0$

Initialize: $w_1 = 0, \bar{g}_0 = 0$.

for $t = 1, 2, 3, \ldots$ **do**

1. Given the function f_t, compute subgradient $g_t \in \partial f_t(w_t)$.
2. Compute the dual average
$$\bar{g}_t = \frac{t-1}{t}\bar{g}_{t-1} + \frac{1}{t}g_t.$$
3. Let $\lambda_t^{\mathrm{RDA}} = \lambda + \gamma\rho/\sqrt{t}$, and compute w_{t+1} entry-wise:
$$w_{t+1}^{(i)} = \begin{cases} 0 & \text{if } \left|\bar{g}_t^{(i)}\right| \leq \lambda_t^{\mathrm{RDA}}, \\ -\dfrac{\sqrt{t}}{\gamma}\left(\bar{g}_t^{(i)} - \lambda_t^{\mathrm{RDA}}\mathrm{sgn}\left(\bar{g}_t^{(i)}\right)\right) & \text{otherwise,} \end{cases} \quad i = 1, \ldots, n. \tag{30}$$

end for

6. Computational Experiments with ℓ_1-Regularization

In this section, we provide computational experiments of the ℓ_1-RDA method on the MNIST data set of handwritten digits (LeCun et al., 1998). Our purpose here is mainly to illustrate the basic characteristics of the ℓ_1-RDA method, rather than comprehensive performance evaluation on a wide range of data sets. First, we describe a variant of the ℓ_1-RDA method that is capable of getting enhanced sparsity in the solution.

6.1 Enhanced ℓ_1-RDA Method

The enhanced ℓ_1-RDA method shown in Algorithm 2 is a special case of Algorithm 1. It is derived by setting $\Psi(w) = \lambda\|w\|_1$, $\beta_t = \gamma\sqrt{t}$, and replacing $h(w)$ with a parameterized version

$$h_\rho(w) = \frac{1}{2}\|w\|_2^2 + \rho\|w\|_1, \tag{31}$$

where $\rho \geq 0$ is a *sparsity-enhancing* parameter. Note that $h_\rho(w)$ is strongly convex with modulus 1 for any $\rho \geq 0$. Hence the convergence rate of this algorithm is the same as if we choose $h(w) = (1/2)\|w\|_2^2$. In this case, the Equation (8) becomes

$$\begin{aligned} w_{t+1} &= \arg\min_w \left\{ \langle \bar{g}_t, w\rangle + \lambda\|w\|_1 + \frac{\gamma}{\sqrt{t}}\left(\frac{1}{2}\|w\|_2^2 + \rho\|w\|_1\right)\right\} \\ &= \arg\min_w \left\{ \langle \bar{g}_t, w\rangle + \lambda_t^{\mathrm{RDA}}\|w\|_1 + \frac{\gamma}{2\sqrt{t}}\|w\|_2^2\right\}, \end{aligned}$$

where $\lambda_t^{\mathrm{RDA}} = \lambda + \gamma\rho/\sqrt{t}$. The above minimization problem has a closed-form solution given in (30) (see Appendix A for the derivation). By letting $\rho > 0$, the effective truncation threshold λ_t^{RDA} is larger than λ, especially in the initial phase of the online process. For problems without explicit ℓ_1-regularization in the objective function, that is, when $\lambda = 0$, this still gives a diminishing truncation threshold $\gamma\rho/\sqrt{t}$.

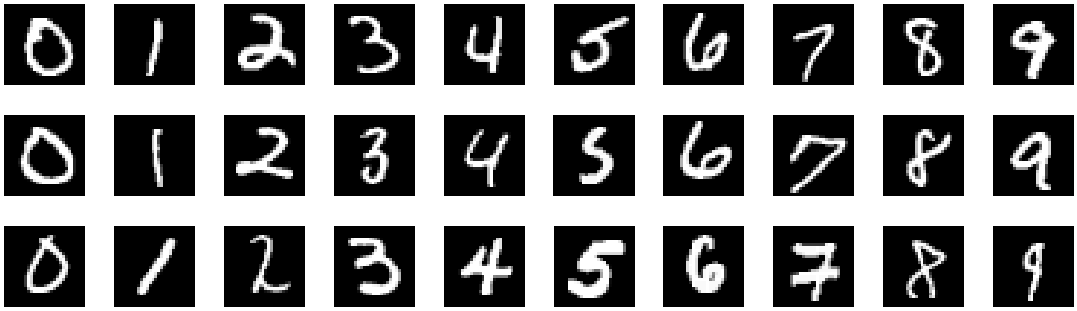


Figure 1: Sample images from the MNIST data set, with gray-scale from 0 to 255.

We can also restrict ℓ_1-regularization on part of the optimization variables only. For example, in support vector machines or logistic regression, we usually want the bias terms to be free of regularization. In this case, we can simply replace λ_t^{RDA} by 0 for the corresponding coordinates in (30).

6.2 Experiments on the MNIST Data Set

Each image in the MNIST data set is represented by a 28×28 gray-scale pixel-map, for a total of 784 features. Each of the 10 digits has roughly 6,000 training examples and 1,000 testing examples. Some of the samples are shown in Figure 1. From the perspective of using stochastic and online algorithms, the number of features and size of the data set are considered very small. Nevertheless, we choose this data set because the computational results are easy to visualize. No preprocessing of the data is employed.

We use ℓ_1-regularized logistic regression to do binary classification on each of the 45 pairs of digits. More specifically, let $z = (x, y)$ where $x \in \mathbf{R}^{784}$ represents a gray-scale image and $y \in \{+1, -1\}$ is the binary label, and let $w = (\tilde{w}, b)$ where $\tilde{w} \in \mathbf{R}^{784}$ and b is the bias. Then the loss function and regularization term in (1) are

$$f(w,z) = \log\left(1 + \exp\left(-y(\tilde{w}^T x + b)\right)\right), \qquad \Psi(w) = \lambda \|\tilde{w}\|_1.$$

Note that we do not apply regularization on the bias term b. In the experiments, we compare the (enhanced) ℓ_1-RDA method (Algorithm 2) with the SGD method

$$w_{t+1}^{(i)} = w_t^{(i)} - \alpha_t \left(g_t^{(i)} + \lambda \operatorname{sgn}(w_t^{(i)})\right), \qquad i = 1, \ldots, n,$$

and the TG method (29) with $\theta = \infty$. These three online algorithms have similar convergence rates and the same order of computational cost per iteration. We also compare them with the batch optimization approach, more specifically solving the empirical minimization problem (2) using an efficient interior-point method (IPM) of Koh et al. (2007).

Each pair of digits have about 12,000 training examples and 2,000 testing examples. We use online algorithms to go through the (randomly permuted) data only once, therefore the algorithms stop at $T = 12{,}000$. We vary the regularization parameter λ from 0.01 to 10. As a reference, the maximum λ for the batch optimization case (Koh et al., 2007) is mostly in the range of $30 - 50$ (beyond which the optimal weights are all zeros). In the ℓ_1-RDA method, we use $\gamma = 5{,}000$, and set ρ to

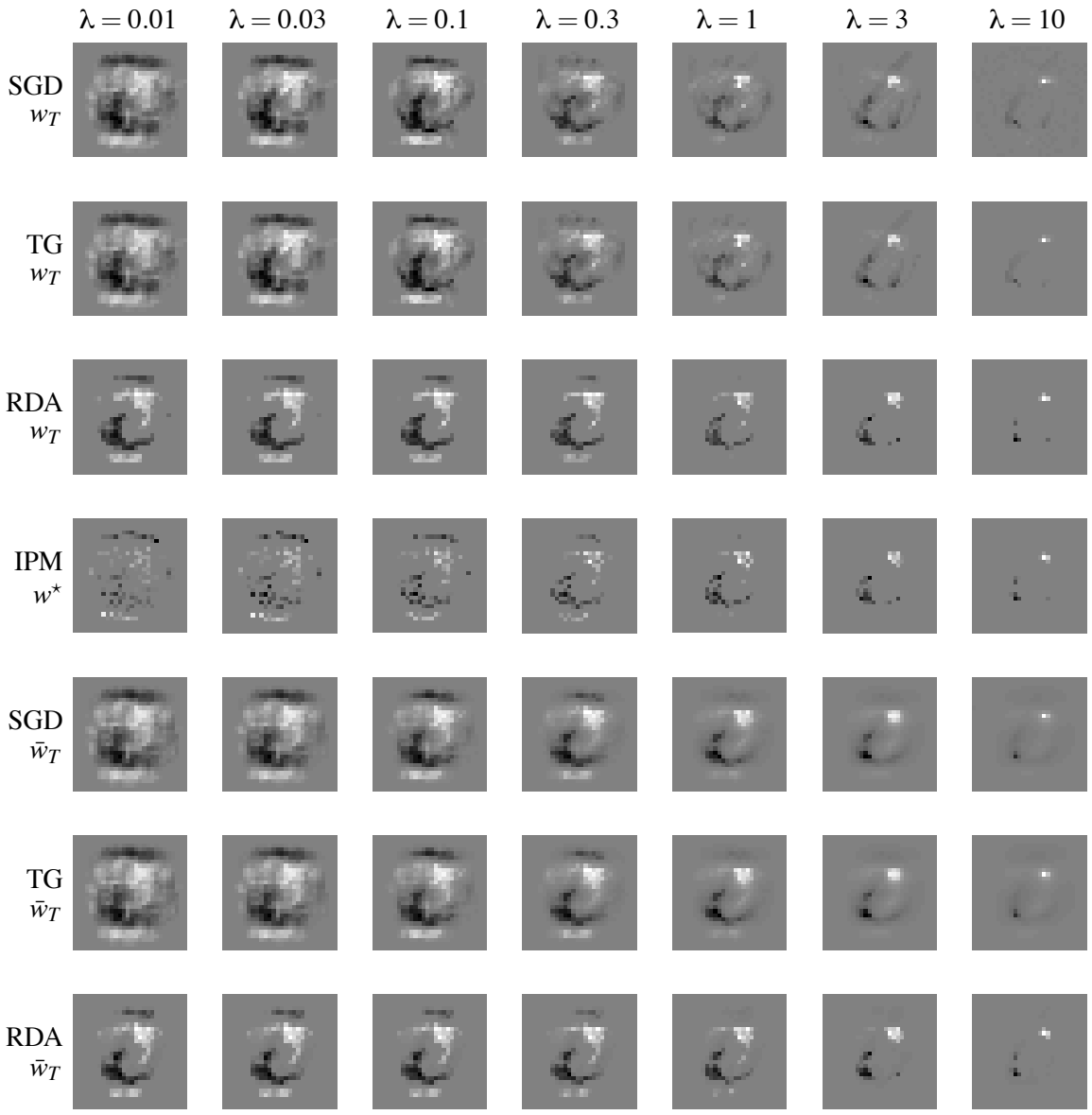


Figure 2: Sparsity patterns of w_T and $\bar{w}_T$ for classifying the digits 6 and 7 when varying the parameter λ from 0.01 to 10 in ℓ_1-regularized logistic regression. The background gray represents the value zero, bright spots represent positive values and dark spots represent negative values. Each column corresponds to a value of λ labeled at the top. The top three rows are the weights w_T (without averaging) from the last iteration of the three online algorithms; the middle row shows optimal solutions of the batch optimization problem solved by interior-point method (IPM); the bottom three rows show the averaged weights $\bar{w}_T$ in the three online algorithms. Both the TG and RDA methods were run with parameters for enhanced ℓ_1-regularization, that is, $K = 10$ for TG and $\gamma\rho = 25$ for RDA.

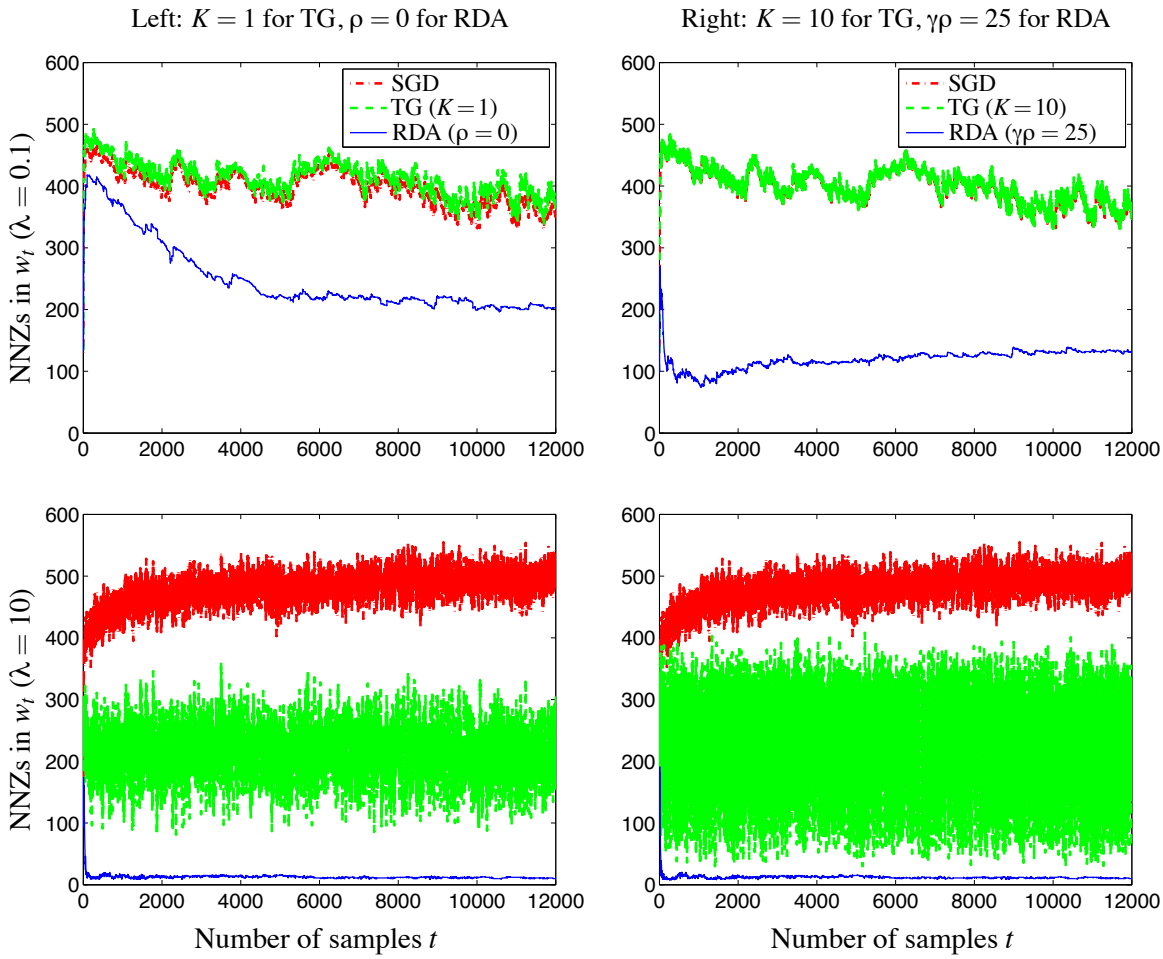


Figure 3: Number of non-zeros (NNZs) in w_t for the three online algorithms (classifying the pair 6 and 7). The left column shows SGD, TG with $K = 1$, and RDA with $\rho = 0$; the right column shows SGD, TG with $K = 10$, and RDA with $\gamma\rho = 25$. The same curves for SGD are plotted in both columns for clear comparison. The two rows correspond to $\lambda = 0.1$ and $\lambda = 10$, respectively.

be either 0 for basic regularization, or 0.005 (effectively $\gamma\rho = 25$) for enhanced regularization effect. These parameters are chosen by cross-validation. For the SGD and TG methods, we use a constant stepsize $\alpha = (1/\gamma)\sqrt{2/T}$ for comparable convergence rate; see (19) and related discussions. In the TG method, the period K is set to be either 1 for basic regularization (same as FOBOS), or 10 for periodic enhanced regularization effect.

Figure 2 shows the sparsity patterns of the solutions w_T and $\bar{w}_T$ for classifying the digits 6 and 7. The algorithmic parameters used are: $K = 10$ for the TG method, and $\gamma\rho = 25$ for the RDA method. It is clear that the RDA method gives more sparse solutions than both SGD and TG methods. The sparsity pattern obtained by the RDA method is very similar to the batch optimization results solved by IPM, especially for larger λ.

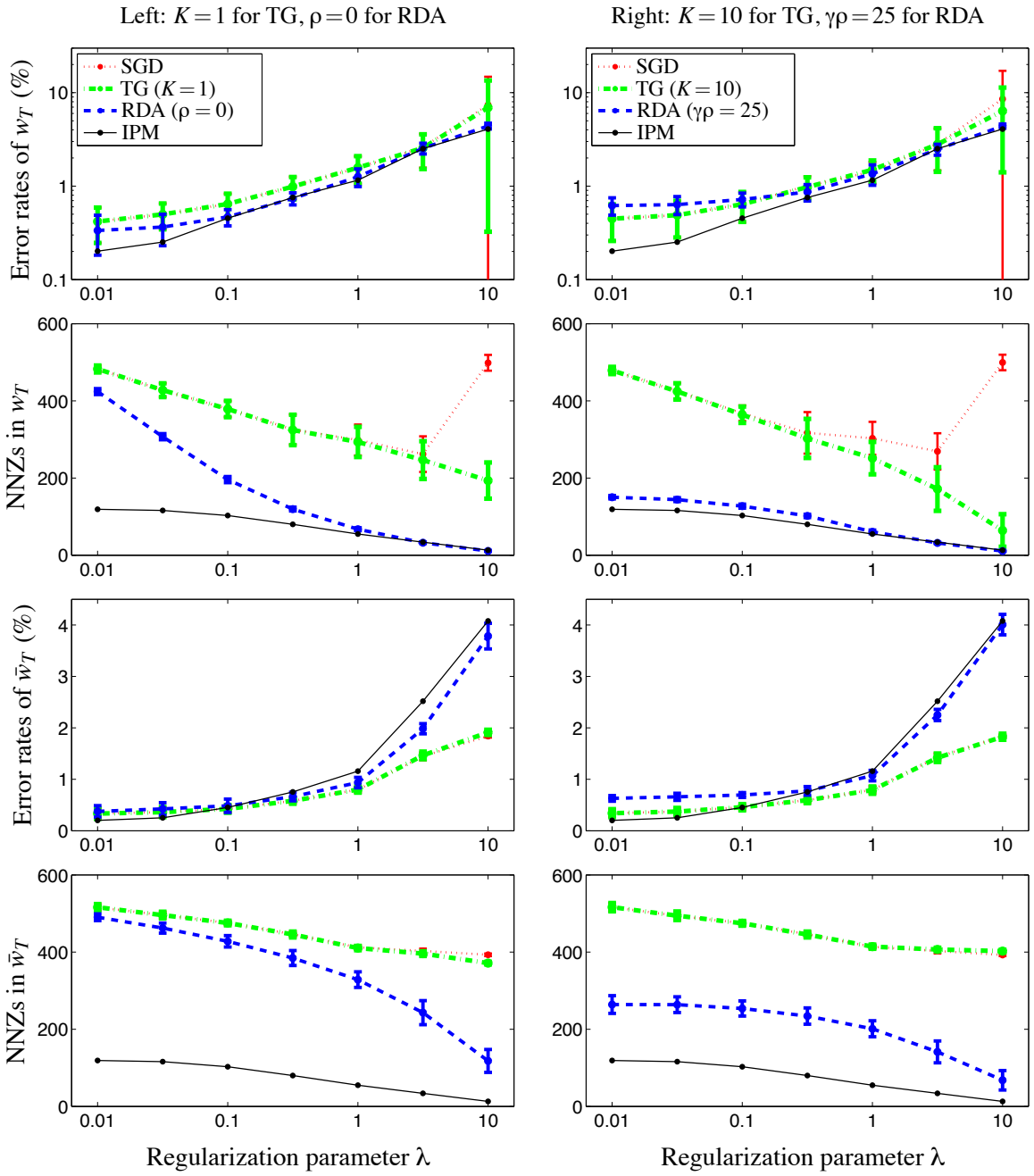


Figure 4: Tradeoffs between testing error rates and NNZs in solutions when varying λ from 0.01 to 10 (for classifying 6 and 7). The left column shows SGD, TG with $K=1$, RDA with $\rho=0$, and IPM. The right column shows SGD, TG with $K=10$, RDA with $\gamma\rho=25$, and IPM. The same curves for SGD and IPM are plotted in both columns for clear comparison. The top two rows shows the testing error rates and NNZs of the final weights w_T, and the bottom two rows are for the averaged weights $\bar{w}_T$. All horizontal axes have logarithmic scale. For vertical axes, only the two plots in the first row have logarithmic scale.

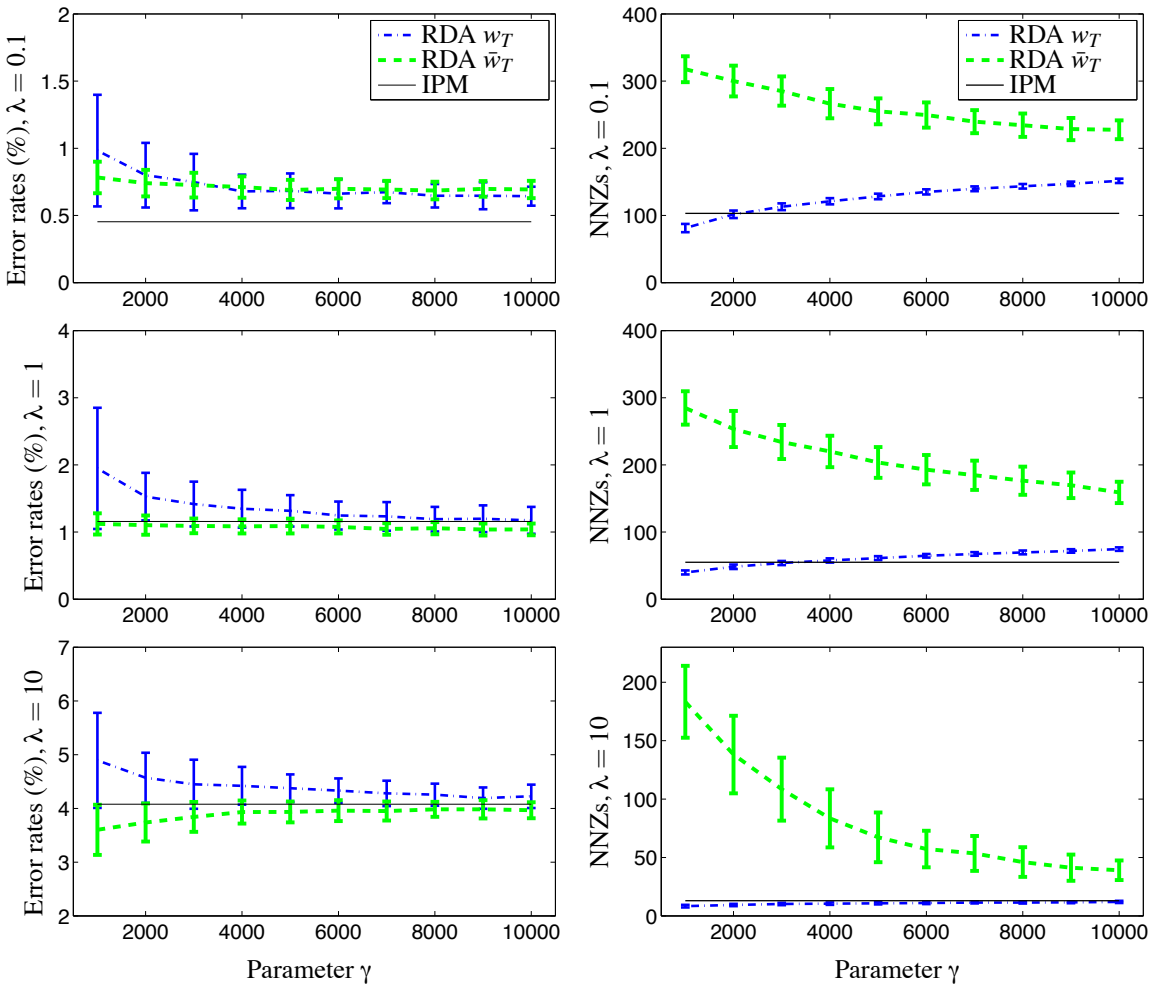


Figure 5: Testing error rates and NNZs in solutions for the RDA method when varying the parameter γ from 1,000 to 10,000, and setting ρ such that $\gamma\rho = 25$. The three rows show results for $\lambda = 0.1$, 1, and 10, respectively. The corresponding batch optimization results found by IPM are shown as a horizontal line in each plot.

To have a better understanding of the behaviors of the algorithms, we plot the number of non-zeros (NNZs) in w_t in Figure 3. Only the RDA method and TG with $K = 1$ give explicit zero weights using soft-thresholding at every step. In order to count the NNZs in all other cases, we have to set a small threshold for rounding the weights to zero. Considering that the magnitudes of the largest weights in Figure 2 are mostly on the order of 10^{-3}, we set 10^{-5} as the threshold and verified that rounding elements less than 10^{-5} to zero does not affect the testing errors. Note that we do not truncate the weights for RDA and TG with $K = 1$ further, even if some of their components are below 10^{-5}. It can be seen that the RDA method maintains a much more sparse w_t than the other online algorithms. While the TG method generates more sparse solutions than the SGD method when λ is large, the NNZs in w_t oscillates with a very big range. The oscillation becomes more severe with $K = 10$. In contrast, the RDA method demonstrates a much more smooth behavior

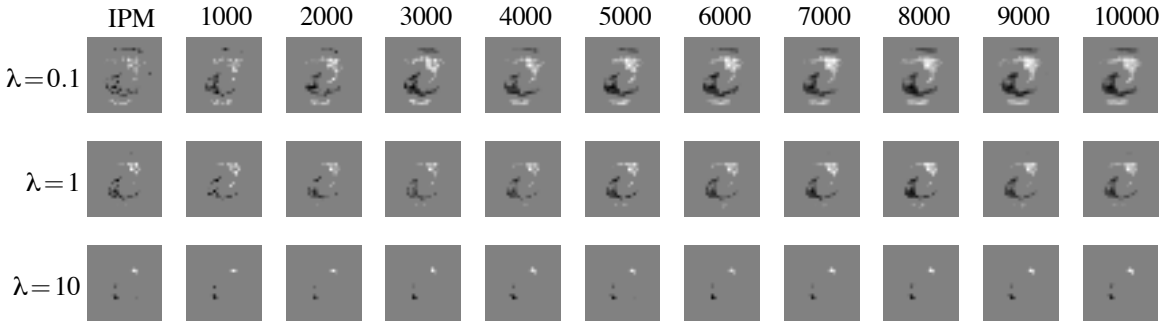


Figure 6: Sparsity patterns of w_T by varying the parameter γ in the RDA method from 1,000 to 10,000 (for classifying the pair 6 and 7). The first column shows results of batch optimization using IPM, and the other 10 columns show results of RDA method using γ labeled at the top.

of the NNZs. For the RDA method, the effect of enhanced regularization using $\gamma\rho = 25$ is more pronounced for relatively small λ.

Next we illustrate the tradeoffs between sparsity and testing error rates. Figure 4 shows that the solutions obtained by the RDA method match the batch optimization results very well. Since the performance of the online algorithms vary when the training data are given in different random sequences (permutations), we run them on 100 randomly permuted sequences of the same training set, and plot the means and standard deviations shown as error bars. For the SGD and TG methods, the testing error rates of w_T vary a lot for different random sequences. In contrast, the RDA method demonstrates very robust performance (small standard deviations) for w_T, even though the theorems only give convergence bound for the averaged weight $\bar{w}_T$. For large values of λ, the averaged weights $\bar{w}_T$ obtained by SGD and TG methods actually have much smaller error rates than those of RDA and batch optimization. This can be explained by the limitation of the SGD and TG methods in obtaining sparse solutions: these lower error rates are obtained with much more nonzero features than used by the RDA and batch optimization methods.

Figure 5 shows the results of choosing different values for the parameter γ in the RDA method. We see that smaller values of γ, which corresponds to faster learning rates, lead to more sparse w_T and higher testing error rates; larger values of γ result in less sparse w_T with lower testing error rates. But interestingly, the effects on the averaged solution $\bar{w}_T$ is almost opposite: smaller values of γ lead to less sparse $\bar{w}_T$ (in this case, we count the NNZs using the rounding threshold 10^{-5}). For large regularization parameter λ, smaller values of γ also give lower testing error rates. Figure 6 shows the sparsity patterns of w_T when varying γ from 1,000 to 10,000. We see that smaller values of γ give more sparse w_T, which are also more scattered like the batch optimization solution by IPM.

Figure 7 shows summary of classification results for all the 45 pairs of digits. For clarity, we only show results of the ℓ_1-RDA method and batch optimization using IPM. We see that the solutions obtained by the ℓ_1-RDA method demonstrate very similar tradeoffs between sparsity and testing error rates as rendered by the batch optimization solutions.

Finally, we note that one of the main reasons for regularization in machine learning is to prevent overfitting, meaning that appropriate amount of regularization may actually reduce the testing error rate. In order to investigate the possibility of overfitting, we also conducted experiments by subsam-

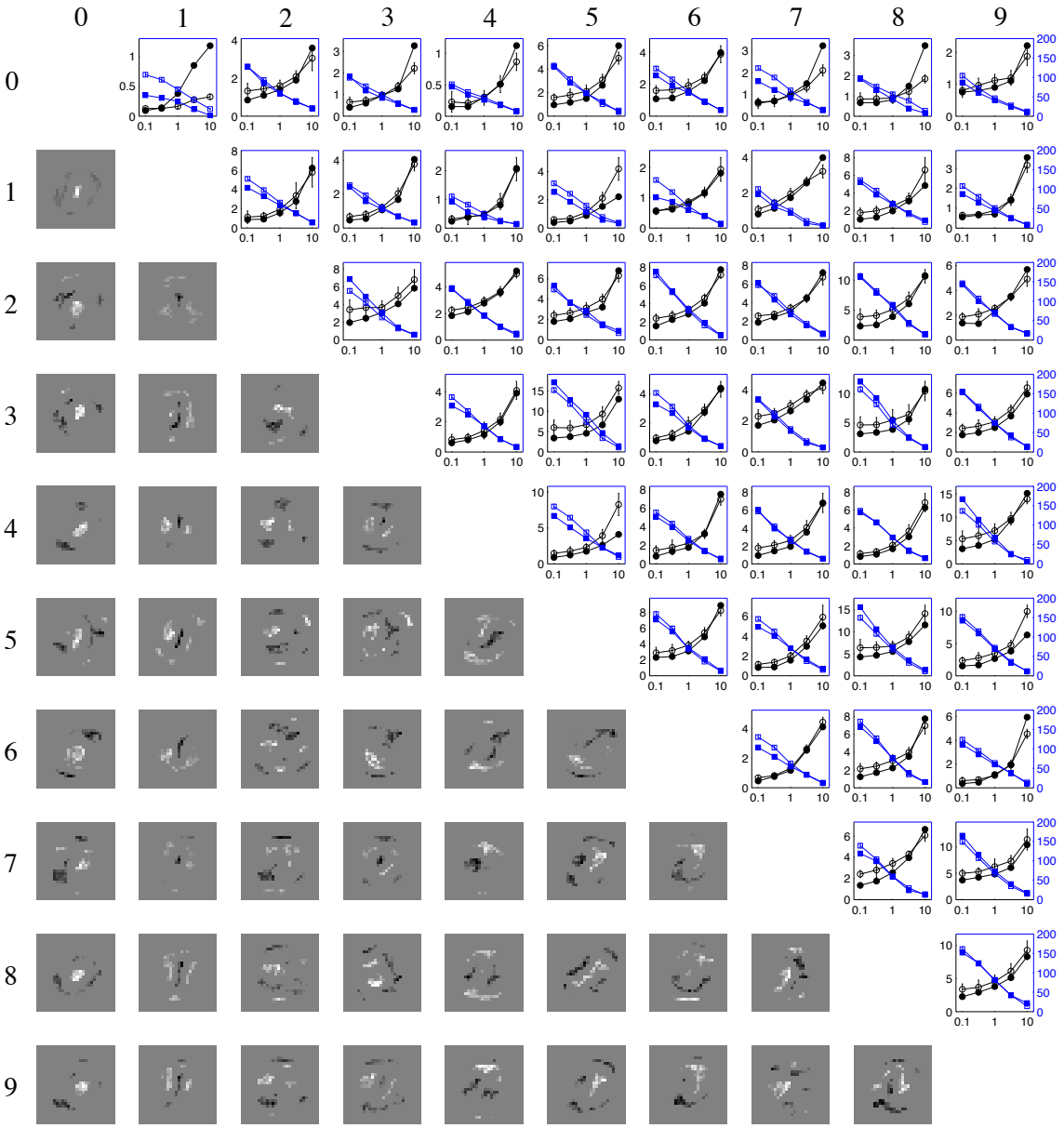


Figure 7: Binary classification for all 45 pairs of digits. The images in the lower-left triangular area show sparsity patterns of w_T with $\lambda = 1$, obtained by the ℓ_1-RDA method with $\gamma = 5000$ and $\rho = 0.005$. The plots in the upper-right triangular area show tradeoffs between sparsity and testing error rates, by varying λ from 0.1 to 10. The solid circles and solid squares show error rates and NNZs in w_T, respectively, using IPM for batch optimization. The hollow circles and hollow squares show error rates and NNZs of w_T, respectively, using the ℓ_1-RDA method. The vertical bars centered at hollow circles and squares show standard deviations by running on 100 different random permutations of the same training data. The scales of the error rates (in percentages) are marked on the left vertical axes, and the scales of the NNZs are marked on the right-most vertical axes.

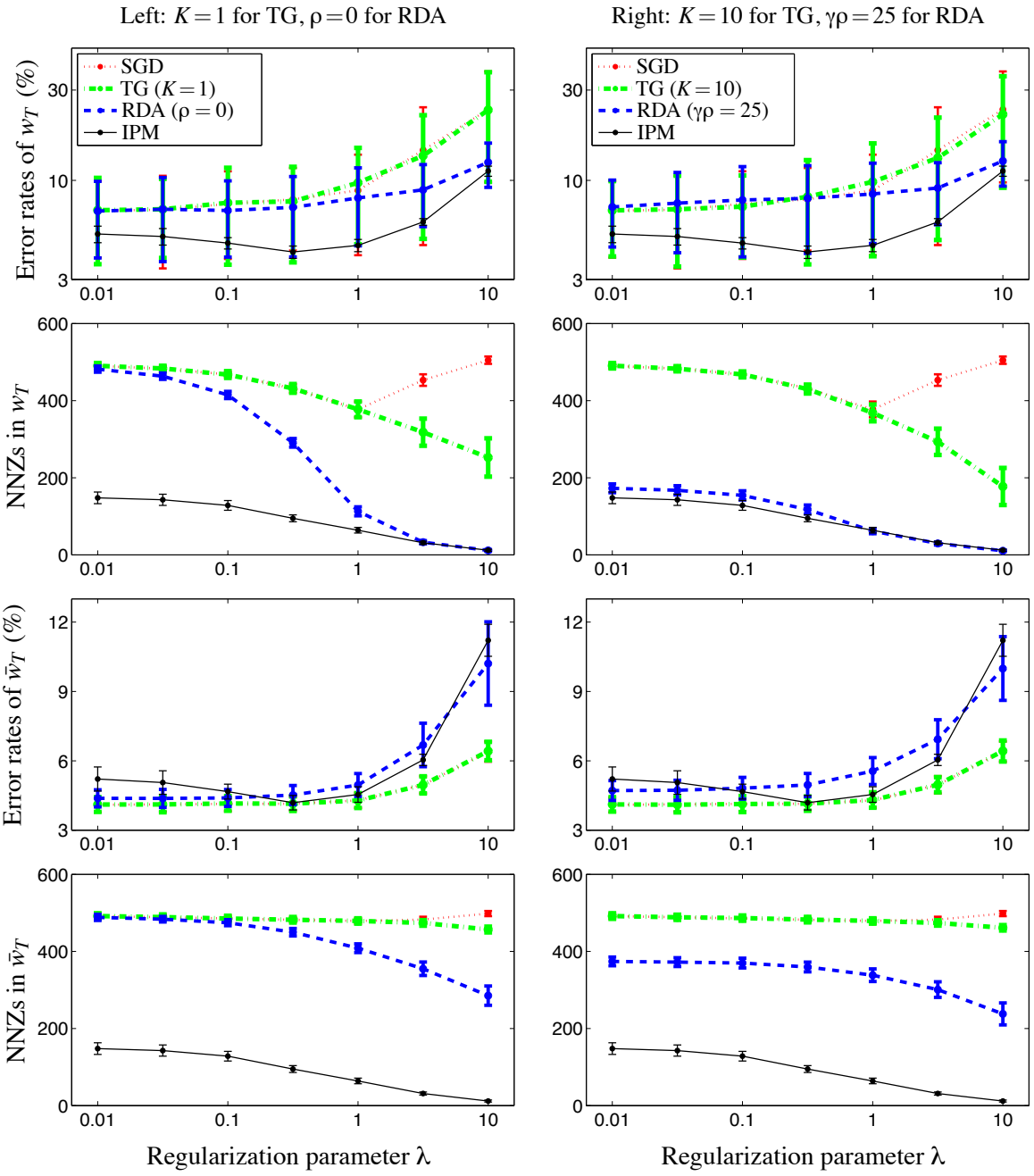


Figure 8: Tradeoffs between testing error rates and NNZs in solutions when varying λ from 0.01 to 10 (for classifying 3 and 8). In order to investigate overfitting, we used 1/10 subsampling of the training data. The error bars show standard deviations of using 10 sets of subsamples. For the three online algorithms, we averaged results on 10 random permutations for each of the 10 subsets. The left column shows SGD, TG with $K=1$, RDA with $\rho=0$, and IPM. The right column shows SGD, TG with $K=10$, RDA with $\gamma\rho=25$, and IPM. The same curves for SGD and IPM are plotted in both columns for clear comparison.

pling the training set. More specifically, we randomly partition the training sets in 10 subsets, and use each subset for training but still test on the whole testing set. The same algorithmic parameters γ and ρ are used as before. Figure 8 shows the results of classifying the more difficult pair 3 and 8. We see that overfitting does occur for batch optimization using IPM. Online algorithms, thanks for their low accuracy in solving the optimization problems, are mostly immune from overfitting.

7. Discussions and Extensions

This paper is inspired by several work in the emerging area of *structural convex optimization* (Nesterov, 2008). The key idea is that by exploiting problem structure that are beyond the conventional black-box model (where only function values and gradient information are allowed), much more efficient first-order methods can be developed for solving structural convex optimization problems. Consider the following problem with two separate parts in the objective function:

$$\underset{w}{\text{minimize}} \quad f(w) + \Psi(w) \tag{32}$$

where the function f is convex and differentiable on $\operatorname{dom}\Psi$, its gradient $\nabla f(w)$ is Lipschitz-continuous with constant L, and the function Ψ is a closed proper convex function. Since Ψ in general can be non-differentiable, the best convergence rate for gradient-type methods that are based on the black-box model is $O(1/\sqrt{t})$ (Nemirovsky and Yudin, 1983). However, if the function Ψ is *simple*, meaning that we are able to find closed-form solution for the auxiliary optimization problem

$$\underset{w}{\text{minimize}} \quad \left\{ f(u) + \langle \nabla f(u), w-u \rangle + \frac{L}{2}\|w-u\|_2^2 + \Psi(w) \right\}, \tag{33}$$

then it is possible to develop accelerated gradient methods that have the convergence rate $O(1/t^2)$ (Nesterov, 1983, 2004; Tseng, 2008; Beck and Teboulle, 2009). Accelerated first-order methods have also been developed for solving large-scale conic optimization problems (Auslender and Teboulle, 2006; Lan et al., 2009; Lu, 2009).

The story is a bit different for stochastic optimization. In this case, the convergence rate $O(1/\sqrt{t})$ cannot be improved in general for convex loss functions with a black-box model. When the loss function $f(w,z)$ have better properties such as differentiability, higher orders of smoothness, and strong convexity, it is tempting to expect that better convergence rates can be achieved. Although these better properties of $f(w,z)$ are inherited by the expected function $\varphi(w) \triangleq \mathbf{E}_z f(w,z)$, almost none of them can really help (Nesterov and Vial, 2008, Section 4). One exception is when the objective function is strongly convex. In this case, the convergence rate for stochastic optimization problems can be improved to $O(\ln t/t)$ (e.g., Nesterov and Vial, 2008), or even $O(1/t)$ (e.g., Polyak and Juditsky, 1992; Nemirovski et al., 2009). For online convex optimization problems, the regret bound can be improved to $O(\ln t)$ (Hazan et al., 2006; Bartlett et al., 2008). But these are still far short of the best complexity result for deterministic optimization with strong convexity assumptions; see, for example, Nesterov (2004, Chapter 2) and Nesterov (2007).

We discuss further the case with a stronger smoothness assumption on the stochastic objective functions. In particular, let $f(w,z)$ be differentiable with respect to w for each z, and the gradient, denoted $g(w,z)$, be Lipschitz continuous. In other words, there exists a constant L such that for any fixed z,

$$\|g(v,z) - g(w,z)\|_* \le L\|v-w\|, \qquad \forall\, v,w \in \operatorname{dom}\Psi. \tag{34}$$

Let $\varphi(w) = \mathbf{E}_z f(w,z)$. Then φ is differentiable and $\nabla\varphi(w) = \mathbf{E}_z g(w,z)$ (e.g., Rockafellar and Wets, 1982). By Jensen's inequality, $\nabla\varphi(w)$ is also Lipschitz continuous with the same constant L. For the regularization function Ψ, we assume there is a constant G_Ψ such that

$$|\Psi(v) - \Psi(w)| \leq G_\Psi \|v - w\|, \qquad \forall\, v, w \in \operatorname{dom}\Psi.$$

In a black-box model, for any query point w, we are only allowed to query a stochastic gradient $g(w,z)$ and a subgradient of $\Psi(w)$. We assume the stochastic gradients have bounded variance; more specifically, let there be a constant Q such that

$$\mathbf{E}_z \|g(w,z) - \nabla\varphi(w)\|_*^2 \leq Q^2, \qquad \forall\, w \in \operatorname{dom}\Psi. \tag{35}$$

Under these assumptions and the black-box model, the optimal convergence rate for solving the problem (1), according to the complexity theory of Nemirovsky and Yudin (1983), is

$$\mathbf{E}\,\phi(w_t) - \phi^\star \leq O(1)\left(\frac{L}{t^2} + \frac{G_\Psi + Q}{\sqrt{t}}\right).$$

Lan (2010) developed an accelerated mirror-descent stochastic approximation method to achieve this rate. The stochastic nature of the algorithm dictates that the term $O(1)(Q/\sqrt{t})$ is inevitable in the convergence bound. However, by using structural optimization techniques similar to (33), it is possible to eliminate the term $O(1)(G_\Psi/\sqrt{t})$ and achieve

$$\mathbf{E}\phi(w_t) - \phi^\star \leq O(1)\left(\frac{L}{t^2} + \frac{Q}{\sqrt{t}}\right). \tag{36}$$

Such a result was obtained by Hu et al. (2009). Their algorithm can be viewed as an accelerated version of the FOBOS method (28). In each iteration of their method, the regularization term $\Psi(w)$ is discounted by a factor of $\Theta(t^{-3/2})$. In terms of obtaining the desired regularization effects (see discussions in Section 5), this is even worse than the $\Theta(t^{-1/2})$ discount factor in the FOBOS method. For the case of ℓ_1-regularization, this means using an even smaller truncation threshold $\Theta(t^{-3/2})\lambda$. Next, we give an accelerated version of the RDA method, which achieves the same improved convergence rate (36), but also maintains the desired property of using the undiscounted regularization at each iteration.

7.1 Accelerated RDA Method for Stochastic Optimization

Nesterov (2005) developed an accelerated version of the dual averaging method for solving smooth convex optimization problems, where the uniform average of all past gradients is replaced by an weighted average that emphasizes more recent gradients. Several variations (Nesterov, 2007; Tseng, 2008) were also developed for minimizing composite objective functions of the form (32). They all have a convergence rate $O(L/t^2)$.

Algorithm 3 is our extension of Nesterov's method for solving stochastic optimization problems of the form (1). At the input, it needs a strongly convex function h and two positive sequences $\{\alpha_t\}_{t\geq 1}$ and $\{\beta_t\}_{t\geq 0}$. At each iteration $t \geq 1$, it computes three primal vectors u_t, v_t, w_t, and a dual vector $\tilde{g}_t$. Among them, u_t is the point for querying a stochastic gradient, $\tilde{g}_t$ is an weighted average of all past stochastic gradients, v_t is the solution of an auxiliary minimization problem that involves $\tilde{g}_t$ and the regularization term $\Psi(w)$, and w_t is the output vector. The computational effort

Algorithm 3 Accelerated RDA method

Input:

- a strongly convex function $h(w)$ with modulus 1 on $\mathrm{dom}\,\Psi$.
- two positive sequences $\{\alpha_t\}_{t\geq 1}$ and $\{\beta_t\}_{t\geq 0}$.

Initialize: set $w_0 = v_0 = \arg\min_w h(w)$, $A_0 = 0$, and $\tilde{g}_0 = 0$.

for $t = 1, 2, 3, \ldots$ **do**

1. Calculate the coefficients

$$A_t = A_{t-1} + \alpha_t, \qquad \theta_t = \frac{\alpha_t}{A_t}.$$

2. Compute the query point

$$u_t = (1-\theta_t) w_{t-1} + \theta_t v_{t-1}.$$

3. Query stochastic gradient $g_t = g(u_t, z_t)$, and update the weighted average $\tilde{g}_t$:

$$\tilde{g}_t = (1-\theta_t)\tilde{g}_{t-1} + \theta_t g_t.$$

4. Solve for the exploration point

$$v_t = \arg\min_w \left\{ \langle \tilde{g}_t, w \rangle + \Psi(w) + \frac{L+\beta_t}{A_t} h(w) \right\}$$

5. Compute w_t by interpolation

$$w_t = (1-\theta_t) w_{t-1} + \theta_t v_t.$$

end for

per iteration is on the same order as Algorithm 1. The additional costs are mainly the two vector interpolations (convex combinations) for computing u_t and w_t. The following theorem gives an estimate of its convergence rate.

Theorem 6 *Assume the conditions (34) and (35) hold, and the problem (1) has an optimal solution $w^\star$ with optimal value $\phi^\star$. In Algorithm 3, if the sequence $\{\alpha_t\}_{t\geq 1}$ and its accumulative sums $A_t = A_{t-1} + \alpha_t$ satisfy the condition $\alpha_t^2 \leq A_t$ for all $t \geq 1$, then*

$$\mathbf{E}\,\phi(w_t) - \phi^\star \leq \frac{L}{A_t} h(w^\star) + \frac{1}{A_t}\left(\beta_t h(w^\star) + Q^2 \sum_{\tau=1}^{t} \frac{\alpha_\tau^2}{2\beta_{\tau-1}} \right).$$

The proof of this theorem is given in Appendix D.

If we choose the two input sequences as

$$\alpha_t = 1, \qquad \forall t \geq 1,$$
$$\beta_t = \gamma\sqrt{t+1}, \qquad \forall t \geq 0,$$

then $A_t = t$, $\theta_t = 1/t$, and $\tilde{g}_t = \bar{g}_t$ is the uniform average of all past gradients. In this case, the minimization problem in Step 4 is very similar to that in Step 3 of Algorithm 1. Let D^2 be an upper

bound on $h(w^\star)$ and set $\gamma = Q/D$. Then we have

$$\mathbf{E}\phi(w_t) - \phi^\star \le \frac{LD^2}{t} + \frac{2QD}{\sqrt{t}}.$$

To achieve the optimal convergence rate stated in (36), we choose

$$\alpha_t = \frac{t}{2}, \qquad \forall t \ge 1, \tag{37}$$

$$\beta_t = \gamma\frac{(t+1)^{3/2}}{2}, \qquad \forall t \ge 0. \tag{38}$$

In this case,

$$A_t = \sum_{\tau=1}^{t}\alpha_\tau = \frac{t(t+1)}{4}, \qquad \theta_t = \frac{\alpha_t}{A_t} = \frac{2}{t+1}, \qquad \forall t \ge 1.$$

It is easy to verify that the condition $\alpha_t^2 \le A_t$ is satisfied. The following corollary is proved in Appendix D.1.

Corollary 7 *Assume the conditions (34) and (35) hold, and $h(w^\star) \le D^2$. If the two input sequences in Algorithm 3 are chosen as in (37) and (38) with $\gamma = Q/D$, then*

$$\mathbf{E}\phi(w_t) - \phi^\star \le \frac{4LD^2}{t^2} + \frac{4QD}{\sqrt{t}}.$$

We can also give high probability bound under more restrictive assumptions. Instead of requiring the deterministic condition $\|g(w,z) - \nabla\varphi(w)\|_*^2 \le Q^2$ for all z and all $w \in \operatorname{dom}\Psi$, we adopt a weaker condition used in Nemirovski et al. (2009) and Lan (2010):

$$\mathbf{E}\left[\exp\left(\frac{\|g(w,z) - \nabla\varphi(w)\|_*^2}{Q^2}\right)\right] \le \exp(1), \qquad \forall w \in \operatorname{dom}\Psi. \tag{39}$$

It is not hard to see that this implies (35) by using Jensen's inequality.

Theorem 8 *Suppose $\operatorname{dom}\Psi$ is compact, say $h(w) \le D^2$ for all $w \in \operatorname{dom}\Psi$, and let the assumptions (34) and (39) hold. If the two input sequences in Algorithm 3 are chosen as in (37) and (38) with $\gamma = Q/D$, then for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\phi(w_t) - \phi^\star \le \frac{4LD^2}{t^2} + \frac{4QD}{\sqrt{t}} + \frac{QD}{\sqrt{t}}\left(\ln(2/\delta) + 2\sqrt{\ln(2/\delta)}\right)$$

Compared with the bound on expectation, the additional penalty in the high probability bound depends only on Q, not L. This theorem is proved in Appendix D.2.

In the special case of deterministic optimization, that is, when $Q = 0$, we have $\gamma = Q/D = 0$ and $\beta_t = 0$ for all $t \ge 0$. Then Algorithm 3 reduces to a variant of Nesterov's method given in Tseng (2008, Section 4), which has convergence rate $\phi(w_t) - \phi^\star \le 4LD^2/t^2$.

For stochastic optimization problems, the above theoretical bounds show that the algorithm can be very effective when Q is much smaller than LD. One way to make this happen is to use a mini-batch approach. More specifically, at each iteration of Algorithm 3, let g_t itself be the average of the stochastic gradients at a small batch of samples computed at u_t. We leave the empirical studies of Algorithm 3 and other accelerated schemes for future investigation.

7.2 The p-Norm RDA Methods

The p-norm RDA methods are special cases of Algorithm 1 in which the auxiliary functions h (*not* the regularization functions Ψ) are squared p-norms. They offer more flexibility than 2-norm based RDA methods in adapting to the geometry of the learning problems.

Recall that for $p \geq 1$, the p-norm of $w \in \mathbf{R}^n$ is defined as $\|w\|_p = \left(\sum_{i=1}^n |w^{(i)}|^p\right)^{1/p}$. If p and q satisfy the equality $1/p + 1/q = 1$, then the norms $\|w\|_p$ and $\|g\|_q$ are dual to each other. Moreover, the pair of functions $(1/2)\|w\|_p^2$ and $(1/2)\|g\|_q^2$ are conjugate functions of each other. As a result, their gradient mappings are a pair of inverse mappings. More formally, let $p \in (1,2]$ and $q = p/(p-1)$, and define the mapping $\vartheta : \mathcal{E} \to \mathcal{E}^*$ with

$$\vartheta_i(w) = \nabla_i \left(\frac{1}{2}\|w\|_p^2\right) = \frac{\operatorname{sgn}\left(w^{(i)}\right)\left|w^{(i)}\right|^{p-1}}{\|w\|_p^{p-2}}, \qquad i = 1, \ldots, n,$$

and the inverse mapping $\vartheta^{-1} : \mathcal{E}^* \to \mathcal{E}$ with

$$\vartheta_i^{-1}(g) = \nabla_i \left(\frac{1}{2}\|g\|_q^2\right) = \frac{\operatorname{sgn}\left(g^{(i)}\right)\left|g^{(i)}\right|^{q-1}}{\|g\|_q^{q-2}}, \qquad i = 1, \ldots, n.$$

These mappings are often called *link functions* in machine learning (e.g., Gentile, 2003).

Again we focus on the ℓ_1-RDA case with $\Psi(w) = \lambda\|w\|_1$. For any $p \in (1,2]$, the function $(1/2)\|w\|_p^2$ is strongly convex with respect to $\|\cdot\|_p$ with the convexity parameter $p-1$ (e.g., Juditsky and Nemirovski, 2008). In order to have an auxiliary strongly convex function h with convexity parameter 1, we define

$$h(w) = \frac{1}{2(p-1)}\|w\|_p^2.$$

Using $\beta_t = \gamma\sqrt{t}$ for some $\gamma > 0$,, the Equation (8) in Algorithm 1 becomes

$$w_{t+1} = \arg\min_w \left\{ \langle \bar{g}_t, w \rangle + \lambda\|w\|_1 + \frac{\gamma}{\sqrt{t}} \frac{1}{2(p-1)}\|w\|_p^2 \right\}.$$

The optimality condition of the above minimization problem (Rockafellar, 1970, Section 27) states that there exists a subgradient $s \in \partial\|w_{t+1}\|_1$ such that

$$\bar{g}_t + \lambda s + \frac{\gamma}{(p-1)\sqrt{t}}\vartheta(w_{t+1}) = 0.$$

Following similar arguments as in Appendix A, we find that it has a closed-form solution

$$w_{t+1} = \vartheta^{-1}\left(\hat{g}_t\right),$$

where the elements of $\hat{g}_t$ are given as

$$\hat{g}_t^{(i)} = \begin{cases} 0 & \text{if } \left|\bar{g}_t^{(i)}\right| \leq \lambda, \\ -\dfrac{(p-1)\sqrt{t}}{\gamma}\left(\bar{g}_t^{(i)} - \lambda \operatorname{sgn}\left(\bar{g}_t^{(i)}\right)\right) & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, n.$$

When $p = q = 2$, ϑ and ϑ^{-1} are identity maps and the solution is the same as (10). If p is close to 1 ($q \gg 2$), the map ϑ^{-1} penalizes small entries of the truncated vector $\hat{g}_t$ to be even smaller.

As an interesting property of the map ϑ^{-1}, we always have $\|w_{t+1}\|_p = \|\hat{g}_t\|_q$ (e.g., Gentile, 2003, Lemma 1).

In terms of regret bound or convergence rate, our results in Sections 3 and 4 apply directly. More specifically, for stochastic learning problems, let $D_p^2 = (1/2(p-1))\|w^\star\|_p^2$, and G_q be an upper bound on $\|g_t\|_q$ for all $t \geq 1$. Then by Corollary 2 and Theorem 3,

$$\mathbf{E}\,\phi(\bar{w}_t) - \phi^\star \leq \left(\gamma D_p^2 + \frac{G_q^2}{\gamma}\right)\frac{1}{\sqrt{t}}.$$

The optimal choice of γ is $\gamma^\star = G_q/D_p$, which results in

$$\mathbf{E}\,\phi(\bar{w}_t) - \phi^\star \leq \sqrt{\frac{2}{p-1}}\frac{G_q\|w^\star\|_p}{\sqrt{t}} = \sqrt{2(q-1)}\frac{G_q\|w^\star\|_p}{\sqrt{t}}.$$

In order to gain further insight, we transform the convergence bound in terms of ℓ_∞ and ℓ_1 norms. Let G_∞ be an upper bound on $\|g_t\|_\infty$, that is,

$$\left|g_t^{(i)}\right| \leq G_\infty, \qquad \forall\, i = 1,\ldots,n, \qquad \forall\, t \geq 1.$$

Then $\|g_t\|_q \leq G_\infty n^{1/q}$. If we choose $q = \ln n$ (assuming $n \geq e^2$ so that $q \geq 2$), then $\|g_t\|_q \leq G_\infty n^{1/\ln n} = G_\infty e$. Next we substitute $G_\infty e$ for G_q and use the fact $\|w^\star\|_p \leq \|w^\star\|_1$, then

$$\mathbf{E}\,\phi(\bar{w}_t) - \phi^\star \leq \sqrt{2(\ln n - 1)}\,\frac{eG_\infty\|w^\star\|_1}{\sqrt{t}} = O\left(\frac{\sqrt{\ln n}\,G_\infty\|w^\star\|_1}{\sqrt{t}}\right).$$

For 2-norm based RDA method, we have $\|g_t\|_2 \leq G_\infty\sqrt{n}$, thus

$$\mathbf{E}\,\phi(\bar{w}_t) - \phi^\star \leq \frac{\sqrt{2n}\,G_\infty\|w^\star\|_2}{\sqrt{t}}.$$

Therefore, for learning problems in which the features are dense (i.e., G_2 close to $G_\infty\sqrt{n}$) and $w^\star$ is indeed very sparse (i.e., $\|w^\star\|_2$ close to $\|w^\star\|_1$), using the p-norm RDA method, with $p = \ln n/(\ln n - 1)$, can lead to faster convergence.

The above analysis of convergence rates matches that for the p-norm based SMIDAS (Stochastic MIrror Descent Algorithm made Sparse) algorithm developed in Shalev-Shwartz and Tewari (2009). However, like other algorithms of the mirror-descent type, including TG (Langford et al., 2009) and FOBOS (Duchi and Singer, 2009), SMIDAS uses a truncation threshold $\Theta(1/\sqrt{t})\lambda$ in obtaining sparse solutions. In contrast, the p-norm based RDA method uses a much more aggressive threshold λ. This is their major difference.

The accelerated RDA method (Algorithm 3) also works in the p-norm setting.

7.3 Connection with Incremental Subgradient Methods

As an intermediate model between deterministic and stochastic optimization problems, consider the problem

$$\underset{w}{\text{minimize}} \quad \sum_{k=1}^{m} f_k(w) + \Psi(w), \tag{40}$$

which can be considered as a special case of (1) where the random variable z has a uniform distribution on a finite support; more specifically, $f_k(w) = (1/m)f(w,z_k)$ for $k = 1,\ldots,m$. The unregularized version, that is, with $\Psi(w) = 0$, has been addressed by *incremental subgradient methods* (e.g., Tseng, 1998; Nedić and Bertsekas, 2001). At each iteration of such methods, a step is taken along the negative subgradient of a single function f_k, which is chosen either in a round-robin manner or randomly with uniform distribution. The randomized version is equivalent to the SGD method. The RDA methods are well suited for solving the regularized version (40).

Randomized incremental subgradient methods with Markov jumps have also been developed for solving (40) with $\Psi(w) = 0$ (Johansson et al., 2009; Ram et al., 2009). In such methods, the functions f_k are picked randomly but not independently: they follow the transition probabilities of a Markov chain that has the uniform distribution. It would be very interesting to investigate the convergence of the RDA methods when the random examples are drawn according to a Markovian chain. This is particularly attractive for online learning problems where the assumption of i.i.d. samples does not hold.

Acknowledgments

The author is grateful to John Duchi for careful reading of a previous draft and pointing out that the regret analysis for general convex regularizations and for strongly convex regularizations can be unified. The author thanks John Platt for encouragement and suggestions for improving the computational experiments, and Paul Tseng and Zhaosong Lu for helpful discussions on minimizing composite objective functions. He also would like to thank the anonymous referees for their valuable comments.

Appendix A. Closed-form Solution for ℓ_1-RDA Method

For RDA method with ℓ_1-regularization, we set $\Psi(w) = \lambda\|w\|_1$ and use $h(w) = (1/2)\|w\|_2^2$, or use $h_\rho(w)$ in (31) for enhanced regularization effect. In such cases, the minimization problem in step 3 of Algorithm 1 can be decomposed into n independent scalar minimization problems, each of the form

$$\underset{\omega\in\mathbf{R}}{\text{minimize}} \quad \eta_t\omega + \lambda_t|\omega| + \frac{\gamma_t}{2}\omega^2,$$

where the coefficients $\lambda_t > 0$, $\gamma_t > 0$, and η_t can be arbitrary. This is an unconstrained nonsmooth optimization problem. Its optimality condition (Rockafellar, 1970, Section 27) states that $\omega^\star$ is an optimal solution if and only if there exists $\xi \in \partial|\omega^\star|$ such that

$$\eta_t + \lambda_t\xi + \gamma_t\omega^\star = 0. \tag{41}$$

The subdifferential of $|\omega|$ is

$$\partial|\omega| = \begin{cases} \{\xi\in\mathbf{R} \mid -1 \le \xi \le 1\} & \text{if } \omega = 0, \\ \{1\} & \text{if } \omega > 0, \\ \{-1\} & \text{if } \omega < 0. \end{cases}$$

We discuss the solution to (41) in three different cases:

- If $|\eta_t| \leq \lambda_t$, then $\omega^\star = 0$ and $\xi = -\eta_t/\lambda_t \in \partial|0|$ satisfy (41). We also show that there is no solution other than $\omega^\star = 0$. If $\omega > 0$, then $\xi = 1$, and

$$\eta_t + \lambda_t + \gamma_t \omega > \eta_t + \lambda_t \geq 0.$$

  Similarly, if $\omega < 0$, then $\xi = -1$, and

$$\eta_t - \lambda_t + \gamma_t \omega < \eta_t - \lambda_t \leq 0.$$

  In either cases when $\omega \neq 0$, the optimality condition (41) cannot be satisfied.

- If $\eta_t > \lambda_t > 0$, we must have $\omega^\star < 0$ and $\xi = -1$. More specifically,

$$\omega^\star = -\frac{1}{\gamma_t}(\eta_t - \lambda_t).$$

- If $\eta_t < -\lambda_t < 0$, we must have $\omega^\star > 0$ and $\xi = 1$. More specifically,

$$\omega^\star = -\frac{1}{\gamma_t}(\eta_t + \lambda_t).$$

The above discussions can be summarized as

$$\omega^\star = \begin{cases} 0 & \text{if } |\eta_t| \leq \lambda_t, \\ -\dfrac{1}{\gamma_t}\big(\eta_t - \lambda_t \operatorname{sgn}(\eta_t)\big) & \text{otherwise.} \end{cases}$$

This is the closed-form solution for each component of w_{t+1} in the ℓ_1-RDA method.

Appendix B. Regret Analysis of RDA Method

In this Appendix, we prove Theorem 1. First, let s_t denote the sum of the subgradients obtained up to time t in the RDA method, that is,

$$s_t = \sum_{\tau=1}^{t} g_\tau = t\bar{g}_t, \tag{42}$$

with the initialization $s_0 = 0$. Then the Equation (8) in Algorithm 1 is equivalent to

$$w_{t+1} = \arg\min_w \big\{ \langle s_t, w \rangle + t\Psi(w) + \beta_t h(w) \big\}. \tag{43}$$

This extends the *simple dual averaging* scheme of Nesterov (2009), where $\Psi(w)$ reduces to the indicator function of a closed convex set. Compared with the analysis in Nesterov (2009), the assumption (7), Lemma 11 and Lemma 12 (below) are new essentials that make the proof work. We also provide refined bounds on the primal and dual variables that relate to the regret with respect to an arbitrary comparison point; see part (b) and (c) of Theorem 1. It seems that the *weighted dual averaging* scheme of Nesterov (2009) cannot be extended when Ψ is a nontrivial regularization function.

B.1 Conjugate Functions and Their Properties

Let w_0 be the unique minimizer of $h(w)$. By the assumption (7), we have

$$w_0 = \arg\min_w h(w) \in \operatorname{Arg\,min}_w \Psi(w).$$

Let $\{\beta_t\}_{t\geq 1}$ be the input sequence to Algorithm 1, which is nonnegative and nondecreasing. In accordance with the assumption (13), we let

$$\beta_0 = \max\{\sigma, \beta_1\} > 0, \tag{44}$$

where σ be the convexity parameter of $\Psi(w)$. For each $t \geq 0$, we define two conjugate-type functions:

$$U_t(s) = \max_{w\in\mathcal{F}_D} \big\{\langle s, w - w_0\rangle - t\Psi(w)\big\}, \tag{45}$$

$$V_t(s) = \max_{w} \big\{\langle s, w - w_0\rangle - t\Psi(w) - \beta_t h(w)\big\}, \tag{46}$$

where $\mathcal{F}_D = \big\{w \in \operatorname{dom}\Psi \,|\, h(w) \leq D^2\big\}$. The maximum in (45) is always achieved because $\mathcal{F}_D$ is a nonempty compact set (which always contains w_0). Because of (44), we have $\sigma t + \beta_t \geq \beta_0 > 0$ for all $t \geq 0$, which means the functions $t\Psi(w) + \beta_t h(w)$ are all strongly convex. Therefore, the maximum in (46) is always achieved, and the maximizer is unique. As a result, we have $\operatorname{dom} U_t = \operatorname{dom} V_t = \mathcal{E}^*$ for all $t \geq 0$. Moreover, by the assumption $\Psi(w_0) = h(w_0) = 0$, both of the functions are nonnegative.

The lemma below is similar to Lemma 2 of Nesterov (2009), but with our new definitions of U_t and V_t. We include the proof for completeness.

Lemma 9 *For any $s \in \mathcal{E}^*$ and $t \geq 0$, we have*

$$U_t(s) \leq V_t(s) + \beta_t D^2.$$

Proof Starting with the definition of $U_t(s)$ and using $\mathcal{F}_D = \{w \in \operatorname{dom}\Psi \,|\, h(w) \leq D^2\}$,

$$\begin{aligned}
U_t(s) &= \max_{w\in\mathcal{F}_D} \{\langle s, w - w_0\rangle - t\Psi(w)\} \\
&= \max_w \min_{\beta\geq 0} \big\{\langle s, w - w_0\rangle - t\Psi(w) + \beta(D^2 - h(w))\big\} \\
&\leq \min_{\beta\geq 0} \max_w \big\{\langle s, w - w_0\rangle - t\Psi(w) + \beta(D^2 - h(w))\big\} \\
&\leq \max_w \big\{\langle s, w - w_0\rangle - t\Psi(w) + \beta_t(D^2 - h(w))\big\} \\
&= V_t(s) + \beta_t D^2.
\end{aligned}$$

For the second equality and the first inequality above, we used standard duality arguments and the max-min inequality; see, for example, Boyd and Vandenberghe (2004, Section 5.4.1). ■

Let $\pi_t(s)$ denote the unique maximizer in the definition of $V_t(s)$; in other words,

$$\begin{aligned}
\pi_t(s) &= \arg\max_w \big\{\langle s, w - w_0\rangle - t\Psi(w) - \beta_t h(w)\big\} \\
&= \arg\min_w \big\{\langle -s, w\rangle + t\Psi(w) + \beta_t h(w)\big\}.
\end{aligned}$$

Comparing with the Equation (43), we have

$$w_{t+1} = \pi_t(-s_t), \qquad \forall t \geq 0.$$

Lemma 10 *The function V_t is convex and differentiable. Its gradient is given by*

$$\nabla V_t(s) = \pi_t(s) - w_0 \tag{47}$$

Moreover, the gradient is Lipschitz continuous with constant $1/(\sigma t + \beta_t)$; that is

$$\|\nabla V_t(s_1) - \nabla V_t(s_2)\| \leq \frac{1}{\sigma t + \beta_t}\|s_1 - s_2\|_*, \qquad \forall s_1, s_2 \in \mathcal{E}^*.$$

Proof Because the function $t\Psi(w) + \beta_t h(w)$ is a strongly convex with convexity parameter $\sigma t + \beta_t$, this lemma follows from classical results in convex analysis; see, for example, Hiriart-Urruty and Lemaréchal (2001, Chapter E, Theorem 4.2.1), or Nesterov (2005, Theorem 1). ■

A direct consequence of Lemma 10 is the following inequality:

$$V_t(s+g) \leq V_t(s) + \langle g, \nabla V_t(s)\rangle + \frac{1}{2(\sigma t + \beta_t)}\|g\|_*^2, \quad \forall s, g \in \mathcal{E}^*. \tag{48}$$

For a proof, see, for example, Nesterov (2004, Theorem 2.1.5).

Lemma 11 *For each $t \geq 1$, we have*

$$V_t(-s_t) + \Psi(w_{t+1}) \leq V_{t-1}(-s_t) + (\beta_{t-1} - \beta_t)h(w_{t+1}).$$

Proof We start with the definition of $V_{t-1}(-s_t)$:

$$\begin{aligned} V_{t-1}(-s_t) &= \max_w \{\langle -s_t, w - w_0\rangle - (t-1)\Psi(w) - \beta_{t-1}h(w)\} \\ &\geq \langle -s_t, w_{t+1} - w_0\rangle - (t-1)\Psi(w_{t+1}) - \beta_{t-1}h(w_{t+1}) \\ &= \big\{\langle -s_t, w_{t+1} - w_0\rangle - t\Psi(w_{t+1}) - \beta_t h(w_{t+1})\big\} + \Psi(w_{t+1}) + (\beta_t - \beta_{t-1})h(w_{t+1}). \end{aligned}$$

Comparing with (43) and (46), we recognize that the expression in the last braces above is precisely $V_t(-s_t)$. Making the substitution and rearranging terms give the desired result. ■

Since by assumption $h(w_{t+1}) \geq 0$ and the sequence $\{\beta_t\}_{t\geq 1}$ is nondecreasing, we have

$$V_t(-s_t) + \Psi(w_{t+1}) \leq V_{t-1}(-s_t), \quad \forall t \geq 2. \tag{49}$$

For $t = 1$, Lemma (11) gives

$$V_1(-s_1) + \Psi(w_2) \leq V_0(-s_1) + (\beta_0 - \beta_1)h(w_2). \tag{50}$$

Since it may happen that $\beta_0 > \beta_1$, we need the following upper bound on $h(w_2)$.

Lemma 12 *Assume* $\max\{\sigma, \beta_1\} > 0$, *and let* $h(w) = (1/\sigma)\Psi(w)$ *if* $\sigma > 0$. *Then*

$$h(w_2) \leq \frac{2\|g_1\|_*^2}{(\beta_1+\sigma)^2}. \tag{51}$$

Proof For $t = 1$, we have $w_1 = w_0, \Psi(w_1) = \Psi(w_0) = 0, h(w_1) = h(w_0) = 0$, and $\bar{g}_1 = g_1$. Since w_2 is the minimizer in (43) for $t = 1$, we have

$$\langle g_1, w_2\rangle + \Psi(w_2) + \beta_1 h(w_2) \leq \langle g_1, w_1\rangle + \Psi(w_1) + \beta_1 h(w_1) = \langle g_1, w_1\rangle.$$

Therefore,

$$\Psi(w_2) + \beta_1 h(w_2) \leq \langle g_1, w_1 - w_2\rangle \leq \|g_1\|_*\|w_2 - w_1\|.$$

On the other hand, by strong convexity of $\Psi(w)$ and $h(w)$, we have

$$\Psi(w_2) + \beta_1 h(w_2) \geq \frac{\sigma+\beta_1}{2}\|w_2 - w_1\|^2.$$

Combining the last two inequalities together, we have

$$\Psi(w_2) + \beta_1 h(w_2) \leq \frac{2\|g_1\|_*^2}{\sigma+\beta_1}.$$

By assumption, if $\sigma = 0$, we must have $\beta_1 > 0$. In this case, since $\Psi(w_2) \geq 0$, we have

$$\beta_1 h(w_2) \leq \Psi(w_2) + \beta_1 h(w_2) \leq \frac{2\|g_1\|_*^2}{\beta_1} \quad \Longrightarrow \quad h(w_2) \leq \frac{2\|g_1\|_*^2}{\beta_1^2} = \frac{2\|g_1\|_*^2}{(\sigma+\beta_1)^2}.$$

If $\sigma > 0$, we have $\Psi(w) = \sigma h(w)$ by assumption, and therefore

$$\Psi(w_2) + \beta_1 h(w_2) = (\sigma+\beta_1)h(w_2) \leq \frac{2\|g_1\|_*^2}{\sigma+\beta_1},$$

which also results in (51). ■

B.2 Bounding the Regret

To measure the quality of the solutions $w_1, \ldots, w_t$, we define the following *gap* sequence:

$$\delta_t = \max_{w\in\mathcal{F}_D}\left\{\sum_{\tau=1}^{t}\big(\langle g_\tau, w_\tau - w\rangle + \Psi(w_\tau)\big) - t\Psi(w)\right\}, \quad t = 1,2,3,\ldots. \tag{52}$$

The gap δ_t is an upper bound on the regret $R_t(w)$ for all $w \in \mathcal{F}_D$. To see this, we use the assumption $w \in \mathcal{F}_D$ and convexity of $f_t(w)$ in the following:

$$\begin{aligned}
\delta_t &\geq \sum_{\tau=1}^{t}\big(\langle g_\tau, w_\tau - w\rangle + \Psi(w_\tau)\big) - t\Psi(w) \\
&\geq \sum_{\tau=1}^{t}\big(f_\tau(w_\tau) - f_\tau(w) + \Psi(w_\tau)\big) - t\Psi(w) \\
&= \sum_{\tau=1}^{t}\big(f_\tau(w_\tau) + \Psi(w_\tau)\big) - \sum_{\tau=1}^{t}\big(f_\tau(w) + \Psi(w)\big) = R_t(w).
\end{aligned} \tag{53}$$

We can also derive an upper bound on δ_t. For this purpose, we add and subtract the sum $\sum_{\tau=1}^{t}\langle g_\tau, w_0\rangle$ in the definition (52), which leads to

$$\delta_t = \sum_{\tau=1}^{t}\big(\langle g_\tau, w_\tau - w_0\rangle + \Psi(w_\tau)\big) + \max_{w\in\mathcal{F}_D}\big\{\langle s_t, w_0 - w\rangle - t\Psi(w)\big\}. \tag{54}$$

We observe that the maximization term in (54) is in fact $U_t(-s_t)$. Therefore, by applying Lemma 9, we have

$$\delta_t \le \sum_{\tau=1}^{t}\big(\langle g_\tau, w_\tau - w_0\rangle + \Psi(w_\tau)\big) + V_t(-s_t) + \beta_t D^2. \tag{55}$$

Next, we show that Δ_t defined in (14) is an upper bound for the right-hand side of the inequality (55). For any $\tau \ge 2$, we have

$$\begin{aligned}
V_\tau(-s_\tau) + \Psi(w_{\tau+1}) &\le V_{\tau-1}(-s_\tau) \\
&= V_{\tau-1}(-s_{\tau-1} - g_\tau) \\
&\le V_{\tau-1}(-s_{\tau-1}) + \langle -g_\tau, \nabla V_{\tau-1}(-s_{\tau-1})\rangle + \frac{\|g_\tau\|_*^2}{2(\sigma(\tau-1)+\beta_{\tau-1})} \\
&= V_{\tau-1}(-s_{\tau-1}) + \langle -g_\tau, w_\tau - w_0\rangle + \frac{\|g_\tau\|_*^2}{2(\sigma(\tau-1)+\beta_{\tau-1})},
\end{aligned}$$

where the four steps above used (49), (42), (48), and (47), respectively. Therefore,

$$\langle g_\tau, w_\tau - w_0\rangle + \Psi(w_{\tau+1}) \le V_{\tau-1}(-s_{\tau-1}) - V_\tau(-s_\tau) + \frac{\|g_\tau\|_*^2}{2(\sigma(\tau-1)+\beta_{\tau-1})}, \quad \forall \tau \ge 2.$$

For $\tau = 1$, we have a similar inequality

$$\langle g_1, w_1 - w_0\rangle + \Psi(w_2) \le V_0(-s_0) - V_1(-s_1) + \frac{\|g_1\|_*^2}{2\beta_0} + (\beta_0 - \beta_1)h(w_2),$$

where the additional term $(\beta_0 - \beta_1)h(w_2)$ comes from using (50). Summing the above inequalities for $\tau = 1,\ldots,t$, and noting that $V_0(-s_0) = V_0(0) = 0$, we arrive at

$$\sum_{\tau=1}^{t}\big(\langle g_\tau, w_\tau - w_0\rangle + \Psi(w_{\tau+1})\big) + V_t(-s_t) \le (\beta_0 - \beta_1)h(w_2) + \frac{1}{2}\sum_{\tau=1}^{t}\frac{\|g_\tau\|_*^2}{\sigma(\tau-1)+\beta_{\tau-1}}.$$

Using $w_1 = w_0 \in \operatorname{Arg\,min}_w \Psi(w)$, we have $\Psi(w_{t+1}) \ge \Psi(w_0) = \Psi(w_1)$. Therefore, adding the nonpositive quantity $\Psi(w_1) - \Psi(w_{t+1})$ to the left-hand side of the above inequality yields

$$\sum_{\tau=1}^{t}\big(\langle g_\tau, w_\tau - w_0\rangle + \Psi(w_\tau)\big) + V_t(-s_t) \le (\beta_0 - \beta_1)h(w_2) + \frac{1}{2}\sum_{\tau=1}^{t}\frac{\|g_\tau\|_*^2}{\sigma(\tau-1)+\beta_{\tau-1}}. \tag{56}$$

Combining the inequalities (53), (55) and (56), and using Lemma 12,

$$R_t(w) \le \delta_t \le \beta_t D^2 + \frac{1}{2}\sum_{\tau=1}^{t}\frac{\|g_\tau\|_*^2}{\sigma(\tau-1)+\beta_{\tau-1}} + \frac{2(\beta_0-\beta_1)\|g_1\|_*^2}{(\beta_1+\sigma)^2}.$$

Finally using the assumption (12) and the definition of Δ_t in (14), we conclude

$$R_t(w) \le \delta_t \le \Delta_t.$$

This proves the regret bound (15).

B.3 Bounding the Primal Variable

We start with the optimality condition for the minimization problem in (43): there exist subgradients $b_{t+1} \in \partial\Psi(w_{t+1})$ and $d_{t+1} \in \partial h(w_{t+1})$ such that

$$\langle s_t + t b_{t+1} + \beta_t d_{t+1},\ w - w_{t+1} \rangle \geq 0, \quad \forall w \in \operatorname{dom}\Psi. \tag{57}$$

By the strong convexity of h and Ψ, we have for any $w \in \operatorname{dom}\Psi$,

$$\Psi(w) \geq \Psi(w_{t+1}) + \langle b_{t+1}, w - w_{t+1} \rangle + \frac{\sigma}{2}\|w_{t+1} - w\|^2, \tag{58}$$

$$h(w) \geq h(w_{t+1}) + \langle d_{t+1}, w - w_{t+1} \rangle + \frac{1}{2}\|w_{t+1} - w\|^2. \tag{59}$$

We multiply both sides of the inequality (58) by t, multiply both sides of the inequality (59) by β_t, and then add them together. This gives

$$\begin{aligned}\frac{1}{2}(\sigma t + \beta_t)\|w_{t+1} - w\|^2 \leq{}& \beta_t h(w) - \beta_t h(w_{t+1}) - \langle t b_{t+1} + \beta_t d_{t+1}, w - w_{t+1} \rangle \\ &+ t\Psi(w) - t\Psi(w_{t+1}).\end{aligned}$$

Using the optimality condition (57), we have

$$\begin{aligned}\frac{1}{2}(\sigma t + \beta_t)\|w_{t+1} - w\|^2 &\leq \beta_t h(w) - \beta_t h(w_{t+1}) + \langle s_t, w - w_{t+1} \rangle + t\Psi(w) - t\Psi(w_{t+1}) \\ &= \beta_t h(w) + \left\{ \langle -s_t, w_{t+1} - w_0 \rangle - t\Psi(w_{t+1}) - \beta_t h(w_{t+1}) \right\} \\ &\quad + t\Psi(w) + \langle s_t, w - w_0 \rangle.\end{aligned}$$

Using (43), we recognize that the collection in the braces is precisely $V_t(-s_t)$. Therefore

$$\frac{1}{2}(\sigma t + \beta_t)\|w_{t+1} - w\|^2 \leq \beta_t h(w) + V_t(-s_t) + t\Psi(w) + \langle s_t, w - w_0 \rangle. \tag{60}$$

Now we expand the last term $\langle s_t, w - w_0 \rangle$ using the definition of s_t:

$$\langle s_t, w - w_0 \rangle = \sum_{\tau=1}^{t} \langle g_\tau, w - w_0 \rangle = \sum_{\tau=1}^{t} \langle g_\tau, w_\tau - w_0 \rangle + \sum_{\tau=1}^{t} \langle g_\tau, w - w_\tau \rangle.$$

By further adding and subtracting $\sum_{\tau=1}^{t} \Psi(w_\tau)$, the right-hand side of (60) becomes

$$\beta_t h(w) + \left\{ V_t(-s_t) + \sum_{\tau=1}^{t} \left(\langle g_\tau, w_\tau - w_0 \rangle + \Psi(w_\tau) \right) \right\} + \sum_{\tau=1}^{t} \langle g_\tau, w - w_\tau \rangle + t\Psi(w) - \sum_{\tau=1}^{t} \Psi(w_\tau).$$

We recognize that the expression in the braces above is exactly the left-hand side in (56). Furthermore, by convexity of f_τ for $\tau \geq 1$,

$$\begin{aligned}\sum_{\tau=1}^{t} \langle g_\tau, w - w_\tau \rangle + t\Psi(w) - \sum_{\tau=1}^{t} \Psi(w_\tau) &\leq \sum_{\tau=1}^{t} \left(f_\tau(w) - f_\tau(w_\tau) \right) + t\Psi(w) - \sum_{\tau=1}^{t} \Psi(w_\tau) \\ &= \sum_{\tau=1}^{t} \left(f_\tau(w) + \Psi(w) \right) - \sum_{\tau=1}^{t} \left(f_\tau(w_\tau) + \Psi(w_\tau) \right) \\ &= -R_t(w).\end{aligned}$$

Putting everything together, and using (56), we have

$$\frac{1}{2}(\sigma t+\beta_t)\|w_{t+1}-w\|^2 \le \beta_t h(w)+(\beta_0-\beta_1)h(w_2)+\frac{1}{2}\sum_{\tau=1}^{t}\frac{\|g_\tau\|_*^2}{\sigma(\tau-1)+\beta_{\tau-1}}-R_t(w).$$

Finally, using $w\in\mathcal{F}_D$, Lemma 12 and the assumption (12), we conclude

$$\frac{1}{2}(\sigma t+\beta_t)\|w_{t+1}-w\|^2 \le \Delta_t - R_t(w),$$

which is the same as (16).

B.4 Bounding the Dual Average

First notice that (54) still holds if we replace w_0 with an arbitrary, fixed $w\in\mathcal{F}_D$, that is,

$$\delta_t=\sum_{\tau=1}^{t}\big(\langle g_\tau, w_\tau-w\rangle+\Psi(w_\tau)\big)+\max_{u\in\mathcal{F}_D}\big\{\langle s_t, w-u\rangle - t\Psi(u)\big\}.$$

By convexity of f_τ for $\tau\ge 1$, we have

$$\begin{aligned}
\delta_t &\ge \sum_{\tau=1}^{t}\big(f_\tau(w_\tau)-f_\tau(w)+\Psi(w_\tau)\big)+\max_{u\in\mathcal{F}_D}\big\{\langle s_t, w-u\rangle - t\Psi(u)\big\}\\
&= \sum_{\tau=1}^{t}\big(f_\tau(w_\tau)+\Psi(w_\tau)-f_\tau(w)-\Psi(w)\big)+\max_{u\in\mathcal{F}_D}\big\{\langle s_t, w-u\rangle - t\big(\Psi(u)-\Psi(w)\big)\big\}\\
&= R_t(w)+\max_{u\in\mathcal{F}_D}\big\{\langle s_t, w-u\rangle - t\big(\Psi(u)-\Psi(w)\big)\big\}. \qquad (61)
\end{aligned}$$

Let $d(u)$ denote a subgradient of Ψ at u with minimum norm, that is,

$$d(u)=\operatorname*{arg\,min}_{g\in\partial\Psi(u)}\|g\|_*. \qquad (62)$$

Since Ψ has convexity parameter σ, we have

$$\Psi(w)-\Psi(u)\ge\langle d(u), w-u\rangle+\frac{\sigma}{2}\|w-u\|^2.$$

Therefore,

$$\begin{aligned}
\delta_t &\ge R_t(w)+\max_{u\in\mathcal{F}_D}\left\{\langle s_t, w-u\rangle + t\langle d(u), w-u\rangle+\frac{\sigma t}{2}\|w-u\|^2\right\}\\
&\ge R_t(w)+\max_{u\in\mathcal{B}(w,r)}\left\{\langle s_t, w-u\rangle + t\langle d(u), w-u\rangle+\frac{\sigma t}{2}\|w-u\|^2\right\},
\end{aligned}$$

where in the last inequality, we used the assumption $\mathcal{B}(w,r)\subset\mathcal{F}_D$ for some $r>0$. Let $u^\star$ be the maximizer of $\langle s_t, w-u\rangle$ within the set $B(w,r)$, that is,

$$u^\star=\operatorname*{arg\,max}_{u\in B(w,r)}\big\{\langle s_t, w-u\rangle\big\}.$$

Then $\|w-u^\star\| = r$ and

$$\langle s_t, w-u^\star\rangle = \|w-u^\star\|\|s_t\|_* = r\|s_t\|_*.$$

So we can continue with the inequality:

$$\begin{aligned}
\delta_t &\geq R_t(w) + \langle s_t, w-u^\star\rangle + t\langle d(u^\star), w-u^\star\rangle + \frac{\sigma t}{2}\|w-u^\star\|^2 \\
&= R_t(w) + r\|s_t\|_* + t\langle d(u^\star), w-u^\star\rangle + \frac{1}{2}\sigma t r^2 \\
&\geq R_t(w) + r\|s_t\|_* - t\|d(u^\star)\|_*\|w-u^\star\| + \frac{1}{2}\sigma t r^2 \\
&\geq R_t(w) + r\|s_t\|_* - rt\Gamma_D + \frac{1}{2}\sigma t r^2
\end{aligned}$$

where in the last inequality, we used $\|d(u^\star)\|_* \leq \Gamma_D$, which is due to (62) and (11). Therefore

$$\|s_t\|_* \leq t\Gamma_D - \frac{1}{2}\sigma t r + \frac{1}{r}\big(\delta_t - R_t(w)\big).$$

Finally, we have (17) by noting $\delta_t \leq \Delta_t$ and $s_t = t\bar{g}$.

Appendix C. Proof of High Probability Bounds

In this Appendix, we prove Theorem 5. First let $\varphi(w) = \mathbf{E}_z f(w,z)$, then by definition $\phi(w) = \varphi(w) + \Psi(w)$. Let $\hat{g}_t$ be the conditional expectation of g_t given w_t, that is,

$$\hat{g}_t = \mathbf{E}[g_t \,|\, w_t] = \mathbf{E}\big[g_t \,|\, \mathbf{z}[t-1]\big].$$

Since $g_t \in \partial f(w_t, z_t)$, we have $\hat{g}_t \in \partial\varphi(w_t)$ (e.g., Rockafellar and Wets, 1982). By the definition of δ_t in (52), for any $w^\star \in \mathcal{F}_D$,

$$\begin{aligned}
\delta_t &\geq \sum_{\tau=1}^{t}\Big(\langle g_\tau, w_\tau - w^\star\rangle + \Psi(w_\tau)\Big) - t\Psi(w^\star) \\
&= \sum_{\tau=1}^{t}\Big(\langle \hat{g}_\tau, w_\tau - w^\star\rangle + \Psi(w_\tau)\Big) - t\Psi(w^\star) + \sum_{\tau=1}^{t}\langle g_\tau - \hat{g}_\tau, w_\tau - w^\star\rangle \\
&\geq \sum_{\tau=1}^{t}\Big(\varphi(w_\tau) - \varphi(w^\star) + \Psi(w_\tau)\Big) - t\Psi(w^\star) + \sum_{\tau=1}^{t}\langle g_\tau - \hat{g}_\tau, w_\tau - w^\star\rangle \\
&= \sum_{\tau=1}^{t}\big(\phi(w_\tau) - \phi(w^\star)\big) + \sum_{\tau=1}^{t}\langle g_\tau - \hat{g}_\tau, w_\tau - w^\star\rangle. \qquad (63)
\end{aligned}$$

where in the second inequality above we used convexity of φ. Now define the random variables

$$\xi_\tau = \langle g_\tau - \hat{g}_\tau, w^\star - w_\tau\rangle, \qquad \forall \tau \geq 1.$$

Combining (63) and the result $\delta_t \leq \Delta_t$ leads to

$$\sum_{\tau=1}^{t}\big(\phi(w_\tau) - \phi(w^\star)\big) \leq \Delta_t + \sum_{\tau=1}^{t}\xi_\tau. \qquad (64)$$

Since w_t is a deterministic function of $\mathbf{z}[t-1]$ and $\hat{g}_t = \mathbf{E}[g_t \,|\, w_t]$, we have

$$\mathbf{E}\big[\xi_\tau \,\big|\, \mathbf{z}[\tau-1]\big] = 0.$$

Therefore the sum $\sum_{\tau=1}^t \xi_\tau$ is a martingale. By the assumptions $h(w_\tau) \le D^2$ and $\|g_\tau\|_* \le L$ for all w_τ, we have

$$\|w - w_\tau\| \le \|w - w_0\| + \|w_\tau - w_0\| \le (2h(w))^{1/2} + (2h(w_\tau))^{1/2} \le 2\sqrt{2}D,$$

and $\|g_\tau - G_\tau\|_* \le \|g_\tau\|_* + \|G_\tau\|_* \le 2L$. Therefore,

$$|\xi_\tau| \le \|g_\tau - G_\tau\|_* \|w - w_\tau\| \le 4\sqrt{2}LD$$

So the sequence of random variables $\{\xi_\tau\}_{\tau=1}^t$ form a bounded martingale difference. Now by Hoeffding-Azuma inequality (Azuma, 1967), we have

$$\text{Prob}\left(\sum_{\tau=1}^t \xi_\tau \ge \Theta\right) \le \exp\left(\frac{-\Theta^2}{2t(4\sqrt{2}LD)^2}\right) = \exp\left(-\frac{\Theta^2}{64L^2D^2t}\right), \qquad \forall\, \Theta > 0.$$

Let $\Omega = \Theta/(8LD\sqrt{t})$, we have

$$\text{Prob}\left(\sum_{\tau=1}^t \xi_\tau \ge 8LD\sqrt{t}\,\Omega\right) \le \exp(-\Omega^2).$$

Now combining with (64) yields

$$\text{Prob}\left(\phi(\bar{w}_t) - \phi^\star \ge \frac{\Delta_t}{t} + \frac{8LD\Omega}{\sqrt{t}}\right) \le \exp(-\Omega^2).$$

Setting $\delta = \exp(-\Omega^2)$ gives the desired result (26).

Appendix D. Convergence Analysis of Accelerated RDA Method

In this appendix, we prove Theorem 6. We will need the following lemma.

Lemma 13 *Let ψ be a closed proper convex function, and h be strongly convex on* $\text{dom}\,\psi$ *with convexity parameter σ_h. If*

$$v = \arg\min_w \big\{\psi(w) + h(w)\big\}, \tag{65}$$

then

$$\psi(w) + h(w) \ge \psi(v) + h(v) + \frac{\sigma_h}{2}\|w - v\|^2, \qquad \forall\, w \in \text{dom}\,\psi.$$

Proof By the optimality condition for (65), there exist $b \in \partial\psi(v)$ and $d \in \partial h(v)$ such that

$$\langle b + d, w - v\rangle \ge 0, \qquad \forall\, w \in \text{dom}\,\psi.$$

Since ψ is convex and h is strongly convex, we have

$$\begin{aligned}
\psi(w) &\ge \psi(v) + \langle b, w - v\rangle, \\
h(w) &\ge h(v) + \langle d, w - v\rangle + \frac{\sigma_h}{2}\|w - v\|^2.
\end{aligned}$$

The lemma is proved by combining the three inequalities above. ■

In Lemma 13, we do not assume differentiability of either ψ or h. Similar results assuming differentiability have appeared in, for example, Chen and Teboulle (1993) and Tseng (2008), where the term $(\sigma_h/2)\|w-v\|^2$ was replaced by the Bregman divergence induced by h.

Our proof combines several techniques appeared separately in Nesterov (2005), Tseng (2008), Lan (2010), and Nemirovski et al. (2009). First, let $\varphi(w) = \mathbf{E}_z f(w,z)$. For every $t \geq 1$, define the following two functions:

$$\ell_t(w) = \varphi(u_t) + \langle \nabla\varphi(u_t), w-u_t\rangle + \Psi(w),$$
$$\hat{\ell}_t(w) = \varphi(u_t) + \langle g_t, w-u_t\rangle + \Psi(w).$$

Note that $\ell_t(w)$ is a lower bound of $\phi(w)$ for all $t \geq 1$. Let $q_t = g_t - \nabla\varphi(u_t)$, then

$$\hat{\ell}_t(w) = \ell_t(w) + \langle q_t, w-u_t\rangle.$$

For each $t \geq 1$, we also define the function

$$\psi_t(w) = \sum_{\tau=1}^{t} \alpha_\tau \hat{\ell}_\tau(w).$$

For convenience, let $\psi_0(w) = 0$. Then step 4 in Algorithm 3 is equivalent to

$$v_t = \arg\min_w \big\{\psi_t(w) + (L+\beta_t)h(w)\big\}. \tag{66}$$

Since $\nabla\varphi$ is Lipschitz continuous with a constant L (see discussions following (34)), we have

$$\varphi(w_t) \leq \varphi(u_t) + \langle \nabla\varphi(u_t), w_t-u_t\rangle + \frac{L}{2}\|w_t-u_t\|^2.$$

Adding $\Psi(w_t)$ to both sides of the above inequality yields

$$\begin{aligned}
\phi(w_t) &\leq \ell_t(w_t) + \frac{L}{2}\|w_t-u_t\|^2 \\
&= \ell_t\big((1-\theta_t)w_{t-1} + \theta_t v_t\big) + \frac{L}{2}\big\|(1-\theta_t)w_{t-1} + \theta_t v_t - u_t\big\|^2 \\
&\leq (1-\theta_t)\ell_t(w_{t-1}) + \theta_t\ell_t(v_t) + \frac{L}{2}\big\|\theta_t v_t - \theta_t v_{t-1}\big\|^2 \\
&= (1-\theta_t)\ell_t(w_{t-1}) + \theta_t\hat{\ell}_t(v_t) - \theta_t\langle q_t, v_t-u_t\rangle + \theta_t^2\frac{L}{2}\big\|v_t-v_{t-1}\big\|^2 \\
&= (1-\theta_t)\ell_t(w_{t-1}) + \frac{1}{A_t}\left(\alpha_t\hat{\ell}_t(v_t) + \frac{\alpha_t^2}{A_t}\frac{L}{2}\big\|v_t-v_{t-1}\big\|^2\right) - \theta_t\langle q_t, v_t-u_t\rangle \\
&\leq (1-\theta_t)\phi(w_{t-1}) + \frac{1}{A_t}\left(\alpha_t\hat{\ell}_t(v_t) + \frac{L}{2}\big\|v_t-v_{t-1}\big\|^2\right) - \theta_t\langle q_t, v_t-u_t\rangle.
\end{aligned}$$

In the second inequality above, we used convexity of ℓ_t and $u_t = (1-\theta_t)w_{t-1} + \theta_t v_{t-1}$, and in the last inequality above, we used $\ell_t(w) \leq \phi(w)$ and the assumption $\alpha_t^2 \leq A_t$. Multiplying both sides of

the above inequality by A_t and noticing $A_t(1-\theta_t) = A_t - \alpha_t = A_{t-1}$, we have

$$\begin{aligned}
A_t\phi(w_t) &\le A_{t-1}\phi(w_{t-1}) + \alpha_t\hat{\ell}_t(v_t) + \frac{L}{2}\left\|v_t - v_{t-1}\right\|^2 - \alpha_t\langle q_t, v_t - u_t\rangle \\
&= A_{t-1}\phi(w_{t-1}) + \alpha_t\hat{\ell}_t(v_t) + \frac{L+\beta_{t-1}}{2}\left\|v_t - v_{t-1}\right\|^2 - \frac{\beta_{t-1}}{2}\left\|v_t - v_{t-1}\right\|^2 \\
&\quad - \alpha_t\langle q_t, v_t - v_{t-1}\rangle - \alpha_t\langle q_t, v_{t-1} - u_t\rangle \\
&\le A_{t-1}\phi(w_{t-1}) + \alpha_t\hat{\ell}_t(v_t) + \frac{L+\beta_{t-1}}{2}\left\|v_t - v_{t-1}\right\|^2 - \frac{\beta_{t-1}}{2}\left\|v_t - v_{t-1}\right\|^2 \\
&\quad + \alpha_t\|q_t\|_*\|v_t - v_{t-1}\| - \alpha_t\langle q_t, v_{t-1} - u_t\rangle.
\end{aligned}$$

Now using the inequality

$$bc - \frac{a}{2}c^2 \le \frac{b^2}{2a}, \qquad \forall a > 0,$$

with $a = \beta_{t-1}$, $b = \alpha_t\|q_t\|_*$, and $c = \|v_t - v_{t-1}\|$, we have

$$A_t\phi(w_t) \le A_{t-1}\phi(w_{t-1}) + \alpha_t\hat{\ell}_t(v_t) + \frac{L+\beta_{t-1}}{2}\left\|v_t - v_{t-1}\right\|^2 + \frac{\alpha_t^2\|q_t\|_*^2}{2\beta_{t-1}} - \alpha_t\langle q_t, v_{t-1} - u_t\rangle.$$

By (66), v_{t-1} is the minimizer of $\psi_{t-1}(v) + (L+\beta_{t-1})h(v)$. Then by Lemma 13, we have

$$\psi_{t-1}(v_t) + (L+\beta_{t-1})h(v_t) \ge \psi_{t-1}(v_{t-1}) + (L+\beta_{t-1})h(v_{t-1}) + \frac{L+\beta_{t-1}}{2}\left\|v_t - v_{t-1}\right\|^2,$$

therefore,

$$\begin{aligned}
A_t\phi(w_t) - \psi_t(v_t) - (L+\beta_{t-1})h(v_t) \le{}& A_{t-1}\phi(w_{t-1}) - \psi_{t-1}(v_{t-1}) - (L+\beta_{t-1})h(v_{t-1}) \\
&+ \frac{\alpha_t^2\|q_t\|_*^2}{2\beta_{t-1}} - \alpha_t\langle q_t, v_{t-1} - u_t\rangle.
\end{aligned}$$

Since $\beta_t \ge \beta_{t-1}$ and $h(v_t) \ge 0$, we can replace the β_{t-1} on the left-hand side with β_t:

$$\begin{aligned}
A_t\phi(w_t) - \psi_t(v_t) - (L+\beta_t)h(v_t) \le{}& A_{t-1}\phi(w_{t-1}) - \psi_{t-1}(v_{t-1}) - (L+\beta_{t-1})h(v_{t-1}) \\
&+ \frac{\alpha_t^2\|q_t\|_*^2}{2\beta_{t-1}} - \alpha_t\langle q_t, v_{t-1} - u_t\rangle.
\end{aligned}$$

Summing the above inequality from $\tau = 1$ to t results in

$$\begin{aligned}
A_t\phi(w_t) \le{}& \psi_t(v_t) + (L+\beta_t)h(v_t) + A_0\phi(w_0) - \psi_0(v_0) - (L+\beta_0)h(v_0) \\
&+ \sum_{\tau=1}^{t}\frac{\alpha_\tau^2\|q_\tau\|_*^2}{2\beta_{\tau-1}} + \sum_{\tau=1}^{t}\alpha_\tau\langle q_\tau, u_\tau - v_{\tau-1}\rangle.
\end{aligned}$$

Using $A_0 = 0, \psi_0(v_0) = 0, h(v_0) = 0$, and (66), we have

$$\begin{aligned}
A_t\phi(w_t) &\le \psi_t(w^\star) + (L+\beta_t)h(w^\star) + \sum_{\tau=1}^{t} \frac{\alpha_\tau^2 \|q_\tau\|_*^2}{2\beta_{\tau-1}} + \sum_{\tau=1}^{t} \alpha_\tau \langle q_\tau, u_\tau - v_{\tau-1}\rangle \\
&= \sum_{\tau=1}^{t} \alpha_\tau \hat{\ell}_\tau(w^\star) + (L+\beta_t)h(w^\star) + \sum_{\tau=1}^{t} \frac{\alpha_\tau^2 \|q_\tau\|_*^2}{2\beta_{\tau-1}} + \sum_{\tau=1}^{t} \alpha_\tau \langle q_\tau, u_\tau - v_{\tau-1}\rangle \\
&= \sum_{\tau=1}^{t} \alpha_\tau \big(\ell_\tau(w^\star) + \langle q_\tau, w^\star - u_\tau\rangle\big) + (L+\beta_t)h(w^\star) + \sum_{\tau=1}^{t} \frac{\alpha_\tau^2 \|q_\tau\|_*^2}{2\beta_{\tau-1}} \\
&\quad + \sum_{\tau=1}^{t} \alpha_\tau \langle q_\tau, u_\tau - v_{\tau-1}\rangle \\
&= \sum_{\tau=1}^{t} \alpha_\tau \ell_\tau(w^\star) + (L+\beta_t)h(w^\star) + \sum_{\tau=1}^{t} \frac{\alpha_\tau^2 \|q_\tau\|_*^2}{2\beta_{\tau-1}} + \sum_{\tau=1}^{t} \alpha_\tau \langle q_\tau, w^\star - v_{\tau-1}\rangle.
\end{aligned}$$

Next, by $\ell_\tau(w^\star) \le \phi(w^\star)$ for all $\tau \ge 1$ and $A_t = \sum_{\tau=1}^{t} \alpha_\tau$, we have

$$A_t\phi(w_t) \le A_t\phi(w^\star) + (L+\beta_t)h(w^\star) + \sum_{\tau=1}^{t} \frac{\alpha_\tau^2 \|q_\tau\|_*^2}{2\beta_{\tau-1}} + \sum_{\tau=1}^{t} \alpha_\tau \langle q_\tau, w^\star - v_{\tau-1}\rangle. \tag{67}$$

Since $\mathbf{E}\big[q_\tau | \mathbf{z}[\tau-1]\big] = 0$ and q_τ is independent of $v_{\tau-1}$, we have $\mathbf{E}\big[\langle q_\tau, w^\star - v_{\tau-1}\rangle | \mathbf{z}[\tau-1]\big] = 0$. Together with the assumption $\mathbf{E}\|q_\tau\|_*^2 \le Q^2$ for all $\tau \ge 1$, we conclude

$$\mathbf{E}\,\phi(w_t) - \phi(w^\star) \le \frac{L+\beta_t}{A_t} h(w^\star) + \frac{1}{A_t}\left(\frac{Q^2}{2} \sum_{\tau=1}^{t} \frac{\alpha_\tau^2}{\beta_{\tau-1}}\right).$$

By rearranging terms on the right-hand side, this finishes the proof for Theorem 6.

D.1 Proof of Corollary 7

Using the two input sequences given in (37) and (38), we have

$$\sum_{\tau=1}^{t} \frac{\alpha_\tau^2}{2\beta_{\tau-1}} = \frac{1}{4\gamma} \sum_{\tau=1}^{t} \tau^{1/2} \le \frac{1}{4\gamma} \int_0^{t+1} \tau^{1/2} d\tau = \frac{(t+1)^{3/2}}{6\gamma}. \tag{68}$$

Plugging them into the conclusion of Theorem 6 gives

$$\mathbf{E}\phi(w_t) - \phi^\star \le \frac{4L}{t(t+1)} h(w^\star) + \frac{(t+1)^{1/2}}{t}\left(2\gamma h(w^\star) + \frac{2Q^2}{3\gamma}\right).$$

Next we use the assumption $h(w^\star) \le D^2$ and let $\gamma = Q/D$. Then

$$\mathbf{E}\phi(w_t) - \phi^\star \le \frac{4LD^2}{t(t+1)} + \frac{(t+1)^{1/2}}{t} \frac{8QD}{3} \le \frac{4LD}{t^2} + \frac{4QD}{\sqrt{t}}.$$

D.2 Proof of Theorem 8

We start with the inequality (67). We will first show high probability bounds for the two summations on the right-hand side of (67) that involve the stochastic quantities q_τ, and then combine them to prove Theorem 8. We need the following result on large-deviation bound for martingales, which can be viewed as an extension to the Hoeffding-Azuma inequality.

Lemma 14 *(Lan et al., 2008, Lemma 6) Let $z_1, z_2, \ldots$ be a sequence of i.i.d. random variables and let $\mathbf{z}[t]$ denote the collection $[z_1, \ldots, z_t]$. If $\xi_t = \xi_t(\mathbf{z}[t])$ are deterministic Borel functions of $\mathbf{z}[t]$ such that the conditional expectations $\mathbf{E}\big[\xi_t \,|\, \mathbf{z}[t-1]\big] = 0$ almost surely and*

$$\mathbf{E}\big[\exp(\xi_t^2/\nu_t^2) \,|\, \mathbf{z}[t-1]\big] \le \exp(1) \tag{69}$$

almost surely, where $\nu_t > 0$ are deterministic. Then for all $t \ge 1$,

$$\text{Prob}\left(\sum_{\tau=1}^{t} \xi_\tau > \Omega \sqrt{\sum_{\tau=1}^{t} \nu_\tau^2}\right) \le \exp\left(-\frac{\Omega^2}{3}\right), \qquad \forall\, \Omega \ge 0.$$

Lemma 15 *Let $\xi_t = \alpha_t \langle q_t, w^\star - v_{t-1}\rangle$. Then for all $t \ge 1$ and any $\Omega > 0$,*

$$\text{Prob}\left(\sum_{\tau=1}^{t} \xi_\tau > \Omega QD \sqrt{\frac{2}{3}(t+1)^3}\right) \le \exp(-\Omega^2/3).$$

Proof Since $\mathbf{E}\big[q_t \,|\, \mathbf{z}[t-1]\big] = 0$ and q_t is independent of $w^\star$ and v_{t-1}, we have

$$\mathbf{E}\big[\xi_t \big| \mathbf{z}[t-1]\big] = \mathbf{E}\big[\alpha_t \langle q_t, w^\star - v_{t-1}\rangle \big| \mathbf{z}[t-1]\big] = 0.$$

Therefore, $\sum_{\tau=1}^{t} \xi_\tau$ is a martingale. By the assumption $(1/2)\|w\|^2 \le h(w) \le D^2$ for all $w \in \text{dom}\,\Psi$, we have $\|w\| \le \sqrt{2}D$ for all $w \in \text{dom}\,\Psi$, and therefore

$$|\xi_t| \le \alpha_t \|q_t\|_* \|w^\star - v_{t-1}\| \le \alpha_t \|q_t\|_* \big(\|w^\star\| + \|v_{t-1}\|\big) \le \alpha_t \|q_t\|_* 2\sqrt{2}D.$$

Using the assumption $\mathbf{E}\big[\exp\big(\|q_t\|_*^2/Q^2\big)\big] \le \exp(1)$, we have

$$\mathbf{E}\left[\exp\left(\frac{\xi_t^2}{(8\alpha_t^2 Q^2 D^2)^2}\right)\Big| \mathbf{z}[t-1]\right] \le \mathbf{E}\left[\exp\left(\frac{(\alpha_t \|q_t\|_* 2\sqrt{2}D)^2}{8\alpha_t^2 Q^2 D^2}\right)\Big| \mathbf{z}[t-1]\right] \le \exp(1).$$

Therefore the condition (69) holds with $\nu_t^2 = 8\alpha_t^2 Q^2 D^2$. We bound $\sum_{\tau=1}^{t} \nu_\tau^2$ as follows:

$$\sum_{\tau=1}^{t} \nu_\tau^2 \le 8Q^2D^2 \sum_{\tau=1}^{t} \alpha_\tau^2 = 2Q^2D^2 \sum_{\tau=1}^{t} \tau^2 \le 2Q^2D^2 \int_0^{t+1} \tau^2 d\tau = \frac{2Q^2D^2}{3}(t+1)^3.$$

Then applying Lemma 14 gives the desired result. ■

Lemma 16 *For all $t \ge 1$ and any $\Lambda > 0$,*

$$\text{Prob}\left(\sum_{\tau=1}^{t} \frac{\alpha_\tau^2}{2\beta_{\tau-1}} \|q_\tau\|_*^2 > (1+\Lambda)\frac{Q^2}{6\gamma}(t+1)^{3/2}\right) \le \exp(-\Lambda).$$

Proof For any given $t \geq 1$, let

$$\Theta_t = \sum_{\tau=1}^{t} \frac{\alpha_\tau^2}{2\beta_{\tau-1}},$$

and

$$\eta_\tau = \frac{\alpha_\tau^2}{2\beta_{\tau-1}} \frac{1}{\Theta_t}, \qquad \tau = 1, \ldots, t.$$

Therefore $\sum_{\tau=1}^{t} \eta_\tau = 1$. By convexity of the function $\exp(\cdot)$,

$$\exp\left(\sum_{\tau=1}^{t} \eta_\tau \frac{\|q_\tau\|_*^2}{Q^2} \right) \leq \sum_{\tau=1}^{t} \eta_\tau \exp\left(\frac{\|q_\tau\|_*^2}{Q^2} \right).$$

Taking expectation and using the assumption (39),

$$\mathbf{E}\exp\left(\sum_{\tau=1}^{t} \eta_\tau \frac{\|q_\tau\|_*^2}{Q^2} \right) \leq \sum_{\tau=1}^{t} \eta_\tau \mathbf{E}\exp\left(\frac{\|q_\tau\|_*^2}{Q^2} \right) \leq \sum_{\tau=1}^{t} \eta_\tau \exp(1) = \exp(1).$$

By Markov's inequality,

$$\mathrm{Prob}\left(\exp\left(\sum_{\tau=1}^{t} \eta_\tau \frac{\|q_\tau\|_*^2}{Q^2} \right) > \exp(1+\Lambda) \right) \leq \frac{\exp(1)}{\exp(1+\Lambda)} = \exp(-\Lambda),$$

which is the same as

$$\mathrm{Prob}\left(\sum_{\tau=1}^{t} \frac{\alpha_\tau^2}{2\beta_{\tau-1}} \|q_\tau\|_*^2 > (1+\Lambda)\Theta_t Q^2 \right) \leq \exp(-\Lambda).$$

Then using the upper bound on Θ_t derived in (68) gives the desired result. ■

Combining Lemma 15, Lemma 16, and the inequality (67), we have

$$\begin{aligned} &\mathrm{Prob}\left(A_t(\phi(w_t) - \phi^\star) > (L+\beta_t)h(w^\star) + (1+\Lambda)\frac{Q^2}{6\gamma}(t+1)^{3/2} + \Omega QD\sqrt{\frac{2}{3}}(t+1)^{3/2} \right) \\ &\quad \leq \exp(-\Lambda) + \exp\left(-\frac{\Omega^2}{3} \right). \end{aligned}$$

Plugging in $A_t = t(t+1)/4$, $\beta_t = (\gamma/2)(t+1)^{3/2}$, and letting $\gamma = Q/D$, $\Omega = \sqrt{3\Lambda}$, we get

$$\mathrm{Prob}\left(\phi(w_t) - \phi^\star > \frac{4LD^2}{t(t+1)} + \left(\frac{8QD}{3} + \frac{2\Lambda QD}{3} + \sqrt{2\Lambda}QD \right) \frac{(t+1)^{1/2}}{t} \right) \leq 2\exp(-\Lambda).$$

Then using the fact $\sqrt{(t+1)/t} \leq \sqrt{2} \leq 3/2$ for all $t \geq 1$ results in

$$\mathrm{Prob}\left(\phi(w_t) - \phi^\star > \frac{4LD^2}{t^2} + \frac{4QD}{\sqrt{t}} + \frac{(\Lambda + 2\sqrt{\Lambda})QD}{\sqrt{t}} \right) \leq 2\exp(-\Lambda).$$

Finally, let $\delta = 2\exp(-\Lambda)$, hence $\Lambda = \ln(2/\delta)$. Then with probability at least $1-\delta$,

$$\phi(w_t) - \phi^\star \leq \frac{4LD^2}{t^2} + \frac{4QD}{\sqrt{t}} + \frac{QD}{\sqrt{t}}\left(\ln(2/\delta) + 2\sqrt{\ln(2/\delta)} \right).$$

This finishes the proof of Theorem 8.

References

G. Andrew and J. Gao. Scalable training of l_1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 33–40, Corvallis, OR, USA, 2007.

A. Auslender and M. Teboulle. Interior gradient and proximal methods for convex and conic optimization. *SIAM Journal on Optimization*, 16:697–725, 2006.

K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19:357–367, 1967.

S. Balakrishnan and D. Madigan. Algorithms for sparse linear classifiers in the massive data setting. *Journal of Machine Learning Research*, 9:313–337, 2008.

P. Bartlett, E. Hazan, and A. Rakhlin. Adaptive online gradient descent. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 65–72. MIT Press, Cambridge, MA, 2008.

A. Beck and M. Teboulle. A fast iterative shrinkage-threshold algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 161–168. MIT Press, Cambridge, MA, 2008.

L. Bottou and Y. LeCun. Large scale online learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 217–224. MIT Press, Cambridge, MA, 2004.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

D. M. Bradley and J. A. Bagnell. Differentiable sparse coding. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 113–120. MIT Press, Cambridge, MA, USA, 2009.

K. Bredies and D. A. Lorenz. Iterated hard shrinkage for minimization problems with sparsity constraints. *SIAM Journal on Scientific Computing*, 30(2):657–683, 2008.

P. Carbonetto, M. Schmidt, and N. De Freitas. An interior-point stochastic approximation method and an l_1-regularized delta rule. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 233–240. MIT Press, 2009.

N. Cesa-Bianchi and G. Lugosi. *Predictioin, Learning, and Games*. Cambridge University Press, 2006.

G. Chen and M. Teboulle. Convergence analysis of a proximal-like minimization algorithm using Bregman functions. *SIAM Journal on Optimization*, 3(3):538–543, August 1993.

G. H.-G. Chen and R. T. Rockafellar. Convergence rates in forward-backward splitting. *SIAM Journal on Optimization*, 7(2):421–444, 1997.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1998.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2873–2898, 2009.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 272–279, 2008.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. To appear in *Journal of Machine Learning Research*, 2010.

M. C. Ferris and T. S. Munson. Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2003.

M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal on Selected Topics in Signal Processing*, 1(4):586–597, 2007.

D. A. Freedman. On tail probabilities for martingales. *The Annals of Probability*, 3(1):100–118, 1975.

C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53:265–299, 2003.

R. Goebel and R. T. Rockafellar. Local strong convexity and local Lipschitz continuity of the gradient of convex functions. *Journal of Convex Analysis*, 15(2):263–270, 2008.

E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of 19th Annual Conference on Computational Learning Theory (COLT)*, pages 499–513, Pittsburgh, PA, USA, 2006.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer, 2001.

C. Hu, J. T. Kwok, and W. Pan. Accelerated gradient methods for stochastic optimization and online learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 781–789. 2009.

B. Johansson, M. Rabi, and M. Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3):1157–1170, 2009.

A. Juditsky and A. Nemirovski. Large deviations of vector-valued martingales in 2-smooth normed spaces. Manuscript submitted to *The Annals of Probability*, 2008. arXiv:0809.0813v1.

A. Juditsky, A. Nazin, A. Tsybakov, and N. Vayatis. Recursive aggregation of estimators by mirror descent algorithm with averaging. *Problems of Information Transmission*, 41(4):368–384, 2005.

S. M. Kakade and A. Tewari. On the generalization ability of online strongly convex programming algorithms. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 801–808. MIT Press, Cambridge, MA, USA, 2009.

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annuals of Mathematical Statistics*, 23:462–466, 1952.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

G. Lan. An optimal method for stochastic composite optimization. To appear in *Mathematical Programming*, 2010.

G. Lan, A. Nemirovski, and A. Shapiro. Validation analysis of robust stochastic approximation methods. Submitted to *Mathematical Programming*, 2008.

G. Lan, Z. Lu, and R. D. C. Monteiro. Primal-dual first-order methods with $O(1/\varepsilon)$ iteration-complexity for cone programming. *Mathematical Programming*, February 2009. Published online, DOI 10.1007/s10107-008-0261-6.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. Dataset available at `http://yann.lecun.com/exdb/mnist`.

P.-L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16:964–979, 1979.

Z. Lu. Primal-dual first-order methods for a class of cone programming with applications to the Dantzig selector. Submitted manuscript, 2009.

A. Nedić and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

A. Nemirovsky and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. J. Wiley & Sons, New York, 1983.

Yu. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Math. Doklady*, 27(2):372–376, 1983. Translated from Russian by A. Rosa.

Yu. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Boston, 2004.

Yu. Nesterov. Smooth minimization of nonsmooth functions. *Mathematical Programming*, 103: 127–152, 2005.

Yu. Nesterov. Gradient methods for minimizing composite objective function. Technical Report 2007/76, Catholic University of Louvain, Center for Operations Research and Econometrics, 2007.

Yu. Nesterov. How to advance in structural convex optimization. *OPTIMA: Mathematical Programming Society Newsletter*, 78:2–5, November 2008.

Yu. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009. Appeared early as CORE discussion paper 2005/67, Catholic University of Louvain, Center for Operations Research and Econometrics.

Yu. Nesterov and J.-Ph. Vial. Confidence level solutions for stochastic programming. *Automatica*, 44(6):1559–1568, 2008.

B. T. Polyak and A. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30:838–855, 1992.

S. Sundhar Ram, A. Nedić, and V. V. Veeravalli. Incremental stochastic subgradient algorithms for convex optimization. *SIAM Journal on Optimization*, 20(2):691–717, 2009.

H. Robbins and S. Monro. A stochastic approximation method. *The Annuals of Mathematical Statistics*, 22:400–407, 1951.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

R. T. Rockafellar and R. J. B. Wets. On the interchange of subdifferentiation and conditional expectation for convex functionals. *Stochastics An International Journal of Probability and Stochastic Processes*, 7(3):173–182, 1982.

S. Shalev-Shwartz and S. M. Kakade. Mind the duality gap: Logarithmic regret algorithms for online optimization. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1457–1464. MIT Press, 2009.

S. Shalev-Shwartz and Y. Singer. Convex repeated games and Fenchel duality. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 1265–1272. MIT Press, 2006.

S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1 regularized loss minimization. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 929–936, Montreal, Canada, 2009.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 807–814, 2007.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.

P. Tseng. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

P. Tseng. A modified forward-backward splitting method for maximal monotone mappings. *SIAM Journal on Control and Optimization*, 38(2):431–446, 2000.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. Manuscript submitted to *SIAM Journal on Optimization*, 2008.

P. Tseng and D. P. Bertsekas. On the convergence of exponential multiplier method for convex programming. *Mathematical Programming*, 60:1–19, 1993.

S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 116–123, Banff, Alberta, Canada, 2004.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 928–936, Washington DC, 2003.

Journal of Machine Learning Research 11 (2010) 2597-2633 Submitted 3/10; Revised 8/10; Published 10/10

Stochastic Composite Likelihood

Joshua V. Dillon JVDILLON@GATECH.EDU
Guy Lebanon LEBANON@CC.GATECH.EDU
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA

Editor: Fernando Pereira

Abstract

Maximum likelihood estimators are often of limited practical use due to the intensive computation they require. We propose a family of alternative estimators that maximize a stochastic variation of the composite likelihood function. Each of the estimators resolve the computation-accuracy trade-off differently, and taken together they span a continuous spectrum of computation-accuracy trade-off resolutions. We prove the consistency of the estimators, provide formulas for their asymptotic variance, statistical robustness, and computational complexity. We discuss experimental results in the context of Boltzmann machines and conditional random fields. The theoretical and experimental studies demonstrate the effectiveness of the estimators when the computational resources are insufficient. They also demonstrate that in some cases reduced computational complexity is associated with robustness thereby increasing statistical accuracy.

Keywords: Markov random fields, composite likelihood, maximum likelihood estimation

1. Introduction

Maximum likelihood estimation is by far the most popular point estimation technique in machine learning and statistics. Assuming that the data consists of n, m-dimensional vectors

$$D = (X^{(1)}, \ldots, X^{(n)}), \quad X^{(i)} \in \mathbb{R}^m, \tag{1}$$

and is sampled iid from a parametric distribution p_{θ_0} with $\theta_0 \in \Theta \subset \mathbb{R}^r$, a maximum likelihood estimator (MLE) $\hat{\theta}_n^{\text{ml}}$ is a maximizer of the log-likelihood function

$$\ell_n(\theta; D) = \sum_{i=1}^{n} \log p_\theta(X^{(i)}), \tag{2}$$
$$\hat{\theta}_n^{\text{ml}} = \arg\max_{\theta \in \Theta} \ell_n(\theta; D).$$

The use of the MLE is motivated by its consistency,[1] that is, $\hat{\theta}_n^{\text{ml}} \to \theta_0$ as $n \to \infty$ with probability 1 (Ferguson, 1996). The consistency property ensures that as the number n of samples grows, the estimator will converge to the true parameter θ_0 governing the data generation process.

An even stronger motivation for the use of the MLE is that it has an asymptotically normal distribution with mean vector θ_0 and variance matrix $(nI(\theta_0))^{-1}$. More formally, we have the

1. The consistency $\hat{\theta}_n^{\text{ml}} \to \theta_0$ with probability 1 is sometimes called strong consistency in order to differentiate it from the weaker notion of consistency in probability $P(|\hat{\theta}_n^{\text{ml}} - \theta_0| < \varepsilon) \to 0$.

following convergence in distribution as $n \to \infty$ (Ferguson, 1996)

$$\sqrt{n}\,(\hat{\theta}_n^{\mathrm{ml}} - \theta_0) \rightsquigarrow N(0, I^{-1}(\theta_0)), \tag{3}$$

where $I(\theta)$ is the $r \times r$ Fisher information matrix

$$I(\theta) = \mathsf{E}_{p_\theta}\{\nabla \log p_\theta(X)(\nabla \log p_\theta(X))^\top\}$$

with ∇f representing the $r \times 1$ gradient vector of $f(\theta)$ with respect to θ. The convergence (3) is especially striking since according to the Cramer-Rao lower bound, the asymptotic variance $(nI(\theta_0))^{-1}$ of the MLE is the smallest possible variance for any estimator. Since it achieves the lowest possible asymptotic variance, the MLE (and other estimators which share this property) is said to be asymptotically efficient.

The consistency and asymptotic efficiency of the MLE motivate its use in many circumstances. Unfortunately, in some situations the maximization or even evaluation of the log-likelihood (2) and its derivatives is impossible due to computational considerations. For instance this is the situation in many high dimensional exponential family distributions, including Markov random fields whose graphical structure contains cycles. This has lead to the proposal of alternative estimators under the premise that a loss of asymptotic efficiency is acceptable—in return for reduced computational complexity.

In contrast to asymptotic efficiency, we view consistency as a less negotiable property and prefer to avoid inconsistent estimators if at all possible. This common viewpoint in statistics is somewhat at odds with recent advances in the machine learning literature promoting non-consistent estimators, for example using variational techniques (Jordan et al., 1999). Nevertheless, we feel that there is a consensus regarding the benefits of having consistent estimators over non-consistent ones.

In this paper, we propose a family of estimators, for use in situations where the computation of the MLE is intractable. In contrast to many previously proposed approximate estimators, our estimators are statistically consistent and admit a precise quantification of both computational complexity and statistical accuracy through their asymptotic variance. Due to the continuous parameterization of the estimator family, we obtain an effective framework for achieving a predefined problem-specific balance between computational tractability and statistical accuracy. We also demonstrate that in some cases reduced computational complexity may in fact act as a regularizer, increasing robustness and therefore accomplishing both reduced computation and increased accuracy. This "win-win" situation conflicts with the conventional wisdom stating that moving from the MLE to pseudo likelihood and other related estimators result in a computational win but a statistical loss. Nevertheless we show that this occurs in some practical situations.

For the sake of concreteness, we focus on the case of estimating the parameters associated with Markov random fields. In this case, we provide a detailed discussion of the accuracy-complexity tradeoff. We include experiments on both simulated and real world data for several models including the Boltzmann machine, conditional random fields, and the Boltzmann linear chain model. Appendix B outlines a road-map of figures and the corresponding exposition.

2. Related Work

There is a large body of work dedicated to tractable learning techniques. Two popular categories are Markov chain Monte Carlo (MCMC) and variational methods. MCMC is a general purpose

technique for approximating expectations and can be used to approximate the normalization term and other intractable portions of the log-likelihood and its gradient (Casella and Robert, 2004). Variational methods are techniques for conducting inference and learning based on tractable bounds (Jordan et al., 1999). A similar approach would be to conduct maximum likelihood estimation for a simpler model that is tractable.

Despite the substantial work on MCMC and variational methods, there are little practical results concerning the convergence and approximation rate of the resulting parameter estimators. Variational techniques are sometimes inconsistent and it is hard to analyze their asymptotic statistical behavior. In the case of MCMC, a number of asymptotic results exist (Casella and Robert, 2004), but since MCMC plays a role inside each gradient descent or EM iteration it is hard to analyze the asymptotic behavior of the resulting parameter estimates. An advantage of our framework is that we are able to directly characterize the asymptotic behavior of the estimator and relate it to the amount of computational savings.

Our work draws on the composite likelihood method for parameter estimation proposed by Lindsay (1988) which in turn generalized the pseudo likelihood of Besag (1974). A selection of more recent studies on pseudo and composite likelihood are Arnold and Strauss (1991), Liang and Yu (2003), Varin and Vidoni (2005), Sutton and McCallum (2007) and Hjort and Varin (2008). Most of the recent studies in this area examine the behavior of the pseudo or composite likelihood in a particular modeling situation. We believe that the present paper is the first to systematically examine statistical and computational tradeoffs in a general quantitative framework. Possible exceptions include the experimental study of texture generation by Zhu and Liu (2002), the work of Xing et al. (2003) which focused on inference rather than parameter estimation, and the examination of generalization performance of small- and large- scale learning systems by Bottou and Bousquet (2008). The work of Liang and Jordan (2008) is also interesting in that the authors employ composite likelihood m-estimators and asymptotic arguments to compare the risk of discriminative and generative models. However, our work differs in theme and technique—we explore the tradeoff between computation and accuracy by way of a fundamentally different estimator.

Composite likelihood techniques, and consequently our work, can be thought of as local contrastive objectives (i.e., pseudo likelihood, contrastive divergence). Vickrey et al. (2010) present a non-local alternative in which the objective is not restricted to using the training label, but rather any assignment.

3. Stochastic Composite Likelihood

In many cases, the absence of a closed form expression for the normalization term prevents the computation of the log-likelihood (2) and its derivatives thereby severely limiting the use of the MLE. A popular example is Markov random fields, wherein the computation of the normalization term is often intractable (see Section 6 for more details). In this paper we propose alternative estimators based on the maximization of a stochastic variation of the composite likelihood.

We denote multiple samples using superscripts and individual dimensions using subscripts. Thus $X_j^{(r)}$ refers to the j-dimension of the r sample. Following standard convention we refer to random variables (RV) using uppercase letters and their corresponding values using lowercase letters. We also use the standard notations for extracting a subset of the dimensions of a random

variable

$$X_S \stackrel{\text{def}}{=} \{X_i : i \in S\}, \qquad X_{-j} \stackrel{\text{def}}{=} \{X_i : i \neq j\}. \tag{4}$$

We start by reviewing the pseudo log-likelihood function (Besag, 1974) associated with the data D (1),

$$p\ell_n(\theta; D) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \sum_{j=1}^{m} \log p_\theta(X_j^{(i)} | X_{-j}^{(i)}). \tag{5}$$

The maximum pseudo likelihood estimator (MPLE) $\hat{\theta}_n^{\text{mpl}}$ is consistent, that is, $\hat{\theta}_n^{\text{mpl}} \to \theta_0$ with probability 1, but possesses considerably higher asymptotic variance than the MLE's $(nI(\theta_0))^{-1}$. Its main advantage is that it does not require the computation of the normalization term as it cancels out in the probability ratio defining conditional distributions

$$p_\theta(X_j | X_{-j}) = p_\theta(X_j | \{X_k : k \neq j\}) = \frac{p_\theta(X)}{\sum_{x_j} p_\theta(X_1, \ldots, X_{j-1}, X_j = x_j, X_{j+1}, \ldots, X_m)}.$$

The MLE and MPLE represent two different ways of resolving the tradeoff between asymptotic variance and computational complexity. The MLE has low asymptotic variance but high computational complexity while the MPLE has higher asymptotic variance but low computational complexity. It is desirable to obtain additional estimators realizing alternative resolutions of the accuracy complexity tradeoff. To this end we define the stochastic composite likelihood whose maximization provides a family of consistent estimators with statistical accuracy and computational complexity spanning the entire accuracy-complexity spectrum.

Stochastic composite likelihood generalizes the likelihood and pseudo likelihood functions by constructing an objective function that is a stochastic sum of likelihood objects. We start by defining the notion of m-pairs and likelihood objects and then proceed to stochastic composite likelihood.

Definition 1 *An m-pair (A,B) is a pair of sets $A,B \subset \{1,\ldots,m\}$ satisfying $A \neq \emptyset = A \cap B$. The likelihood object associated with an m-pair (A,B) and X is $S_\theta(A,B) \stackrel{\text{def}}{=} \log p_\theta(X_A | X_B)$ where X_S is defined in* (4)*. The composite log-likelihood function (Lindsay, 1988) is a collection of likelihood objects defined by a finite sequence of m-pairs $(A_1,B_1),\ldots,(A_k,B_k)$*

$$c\ell_n(\theta; D) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \sum_{j=1}^{k} \log p_\theta(X_{A_j}^{(i)} | X_{B_j}^{(i)}). \tag{6}$$

There is a certain lack of flexibility associated with the composite likelihood framework as each likelihood object is either selected or not for the entire sample $X^{(1)},\ldots,X^{(n)}$. There is no allowance for some objects to be selected more frequently than others. For example, available computational resources may allow the computation of the log-likelihood for 20% of the samples, and the pseudo likelihood for the remaining 80%. In the case of composite likelihood if we select the full likelihood component (or the pseudo likelihood or any other likelihood object) then this component is applied to all samples indiscriminately.

In SCL, different likelihood objects $S_\theta(A_j,B_j)$ may be selected for different samples with the possibility of some likelihood objects being selected for only a small fraction of the data samples.

The selection may be non-coordinated, in which case each component is selected or not independently of the other components. Or it may be coordinated in which case the selection of one component depends on the selection of the other ones. For example, we may wish to avoid selecting a pseudo likelihood component for a certain sample $X^{(i)}$ if the full likelihood component was already selected for it.

Another important advantage of stochastic selection is that the discrete parameterization of (6) defined by the sequence $(A_1,B_1),\ldots,(A_k,B_k)$ is less convenient for theoretical analysis. Each component is either selected or not, turning the problem of optimally selecting components into a hard combinatorial problem. The stochastic composite likelihood, which is defined below, enjoys continuous parameterization leading to more convenient optimization techniques and convergence analysis.

Definition 2 *Consider a finite sequence of m-pairs* $(A_1,B_1),\ldots,(A_k,B_k)$*, a data set* $D=(X^{(1)},\ldots,X^{(n)})$*,* $\beta\in\mathbb{R}_+^k$*, and n iid, length k, binary random vectors* $Z^{(1)},\ldots,Z^{(n)}\overset{\text{i.i.d.}}{\sim}P(Z)$ *with* $\lambda_j\overset{\text{def}}{=}E(Z_j)>0$*. The stochastic composite log-likelihood (SCL) is*

$$sc\ell_n(\theta;D,Z)\overset{\text{def}}{=}\frac{1}{n}\sum_{i=1}^{n}m_\theta(X^{(i)},Z^{(i)}),\quad \textit{where}$$

$$m_\theta(X,Z)\overset{\text{def}}{=}\sum_{j=1}^{k}\beta_j Z_j\log p_\theta(X_{A_j}|X_{B_j}), \tag{7}$$

where, for brevity, we typically omit D,Z *in favor of* $sc\ell_n(\theta)$*.*

In other words, the SCL is a stochastic extension of (6) where for each sample $X^{(i)},i=1,\ldots,n$, the likelihood objects $S(A_1,B_1),\ldots,S(A_k,B_k)$ are either selected or not, depending on the values of the binary random variables $Z_1^{(i)},\ldots,Z_k^{(i)}$ and weighted by the constants $\beta_1,\ldots,\beta_k$. Note that $Z_j^{(i)}$ may in general depend on $Z_r^{(i)}$ but not on $Z_r^{(l)}$ or on $X^{(i)}$.

When we focus on examining different models for $P(Z)$ we sometimes parameterize it, for example by λ, that is, $P_\lambda(Z)$. This reuse of λ (it is also used in Definition 2) is a notational abuse. We accept it, however, as in most of the cases that we consider $\lambda_1,\ldots,\lambda_k$ from Definition 2 either form the parameter vector for $P(Z)$ or are part of it. Often we refer to a particular λ as a "policy" in order to emphasize its role as a "knob" in selecting particular m-pairs.

Some illustrative examples follow.

Independence. Factorizing $P_\lambda(Z_1,\ldots,Z_k)=\prod_j P_{\lambda_j}(Z_j)$ leads to $Z_j^{(i)}\sim\text{Ber}(\lambda_j)$ with complete independence among the indicator variables. For each sample $X^{(i)}$, each likelihood object $S(A_j,B_j)$ is selected or not independently with probability λ_j.

Multinomial. A multinomial model $Z\sim\text{Mult}(1,\lambda)$ implies that for each sample $Z^{(i)}$ a multivariate Bernoulli experiment is conducted with precisely one likelihood object being selected depending on the selection probabilities $\lambda_1,\ldots,\lambda_k$.

Product of Multinomials. A product of multinomials is formed by a partition of the dimensions to l disjoint subsets $\{1,\ldots,m\}=C_1\cup\cdots C_l$ where $Z_{C_i}\sim\text{Mult}(1,(\lambda_j:j\in C_i))$, that is,

$$P(Z)=\prod_{i=1}^{c}P_i\left(\{Z_j:j\in C_i\}\right),\qquad\text{where } P_i \text{ is } \text{Mult}(1,(\lambda_j:j\in C_l)).$$

Loglinear Models. The distribution $P(Z)$ follows a hierarchical loglinear model (Bishop et al., 1975). This case subsumes the other cases above.

In analogy to the MLE and the MPLE, the maximum SCL estimator (MSCLE) $\hat{\theta}_n^{\text{msl}}$ estimates θ_0 by maximizing the SCL function. In contrast to the log-likelihood and pseudo log-likelihood functions, the SCL function and its maximizer are random variables that depend on the indicator variables $Z^{(1)},\ldots,Z^{(n)}$ in addition to the data D. As such, its behavior should be summarized by examining the limit $n\to\infty$. Doing so eliminates the dependency on particular realizations of $Z^{(1)},\ldots,Z^{(n)}$ in favor of the the expected frequencies $\lambda_j = \mathsf{E}_{P(Z)}Z_j$ which are non-random constants.

The statistical accuracy and computational complexity of the SCL estimator are continuous functions of the parameters (β,λ) (component weights and selection probabilities respectively) which vary continuously throughout their domain $(\lambda,\beta)\in\Lambda\times\mathbb{R}_+^k$. Choosing appropriate values of (λ,β) retrieves the special cases of MLE, MPLE, maximum composite likelihood with each selection being associated with a distinct statistical accuracy and computational complexity. The SCL framework allows selections of many more values of (λ,β) realizing a wide continuous spectrum of estimators, each resolving the accuracy-complexity tradeoff differently.

We include below a demonstration of the SCL framework in a simple low dimensional case. In the following sections we discuss in detail the statistical behavior of the MSCLE and its computational complexity. We conclude the paper with several experimental studies.

3.1 Boltzmann Machine Example

Before proceeding we illustrate the SCL framework using a simple example involving a Boltzmann machine (Hinton and Sejnowski, 1983). Section 8.1 describes the specifics of this model. We consider in detail three SCL policies: full likelihood (FL), pseudo likelihood (PL), and a stochastic combination of first and second order pseudo likelihood with the first order components $p(X_i|X_{-i})$ selected with probability λ and the second order components $p(X_i,X_j|X_{\{i,j\}^c})$ selected with probability $1-\lambda$.

Denoting the number of (binary) graph vertices, or nodes, by m, and the number of examples by n, the computational complexity of the FL function measured in FLOP[2] counts is $O\big(\binom{m}{2}(2^m+n)\big)$ (for the log-likelihood) and $O\big(\binom{m}{2}^2 2^m + n\binom{m}{2}\big)$ for the log-likelihood gradient.[3] The exponential growth in m prevents such computations for large graphs.

The k-order PL function offers a practical alternative to FL (1-order PL corresponds to the traditional pseudo likelihood and 2-order PL its analog, $p(X_{\{i,j\}}|X_{\{i,j\}^c})$). The complexity of computing the corresponding SCL function is $O\big(\binom{m}{2}(\binom{m}{k}2^k+n)\big)$ for the objective function and $O\big(\binom{m}{2}^2\binom{m}{k}2^k + n\binom{m}{2}\big)$ for its gradient. The slower complexity growth of the k-order PL (polynomial in m instead of exponential) is offset by its reduced statistical accuracy, which we measure using the normalized asymptotic variance

$$\text{eff}(\hat{\theta}_n) = \frac{\det(\text{Asymp Var}(\hat{\theta}_n))}{\det(\text{Asymp Var}(\hat{\theta}_n^{\text{ml}}))} \tag{8}$$

2. FLOP is the number of FLoating point OPerations.
3. With memoization the complexity of the gradient can be reduced to $O\big(\binom{m}{2}2^m + n\binom{m}{2}\big)$ (at the cost of exponential 2^m storage). Note that this is only a polynomial improvement to an exponential complexity hence we lose no insight by making naive assumptions.

which is bounded from below by 1 (due to Cramer Rao lower bound) and its deviation from 1 reflects its inefficiency relative to the MLE.

The MLE thus achieves the best accuracy but it is computationally intractable. The first order and second order PL have higher asymptotic variance but are easier to compute. The SCL framework enables adding many more estimators filling in the gaps between ML, 1-order PL, 2-order PL, etc.

We illustrate three SCL functions in the context of a simple Boltzmann machine (five binary nodes, fourteen samples $X^{(1)},\ldots,X^{(14)}$, $\theta^{\text{true}} = (-1,-1,-1,-1,-1,1,1,1,1,1))$ in Figure 1. The top box refers to the full likelihood policy, that is, maximum likelihood. For each of the fourteen samples, the FL component is computed and their aggregation forms the SCL function which in this case equals the log-likelihood. The selection of the FL component for each sample is illustrated using a diamond box. The values under the boxes reflect the FLOP counts needed to compute the components and the total complexity associated with computing the entire SCL or log-likelihood is listed on the right. As mentioned above, the normalized asymptotic variance (8) is 1.

The pseudo likelihood function (5) is illustrated in the second box where each row correspond to one of the five PL components. As each of the five PL component is selected for each of the samples we have diamond boxes covering the entire 5×14 array. The shade of the diamond boxes reflects the complexity required to compute them enabling an easy comparison to the FL components in the top of the figure (note how the FL boxes are much darker than the PL boxes). The numbers at the bottom of each column reflect the FLOP marginal count for each of the fourteen samples and the numbers to the right of the rows reflect the FLOP marginal count for each of the PL components. In this case the FLOP count is less than half the FLOP count of the FL in top box (this reduction in complexity obtained by replacing FL with PL will increase dramatically for graphs with more than 5 nodes) but the asymptotic variance is 83% higher.[4]

The third SCL policy reflects a stochastic combination of first and second order pseudo likelihood components. Each first order component is selected with probability λ and each second order component is selected with probability $1-\lambda$. The result is a collection of 5 1-order PL components and 10 2-order components with only some of them selected for each of the fourteen samples. Again diamond boxes correspond to selected components which are shaded according to their FLOP complexity. The per-component FLOP marginals and per example FLOP marginals are listed as the bottom row and right-most column. The total complexity is somewhere between FL and PL and the asymptotic variance is reduced from the PL's 183% to 148%.

Additional insight may be gained at this point by considering Figure 3 which plots several SCL estimators as points in the plane whose x and y coordinates correspond to normalized asymptotic variance and computational complexity respectively. We turn at this point to considering the statistical properties of the SCL estimators.

4. Consistency and Asymptotic Variance of $\hat{\theta}_n^{\text{msl}}$

A nice property of the SCL framework is enabling mathematical characterization of the statistical properties of the estimator $\hat{\theta}_n^{\text{msl}}$. In this section we examine the conditions for consistency of the MSCLE and its asymptotic distribution and in the next section we consider robustness. The propositions below constitute novel generalizations of some well-known results in classical statistics. Proofs may be found in Appendix A. For simplicity, we assume that X is discrete and $p_\theta(x) > 0$.

4. The asymptotic variance of SCL functions is computed using formulas derived in the next section.

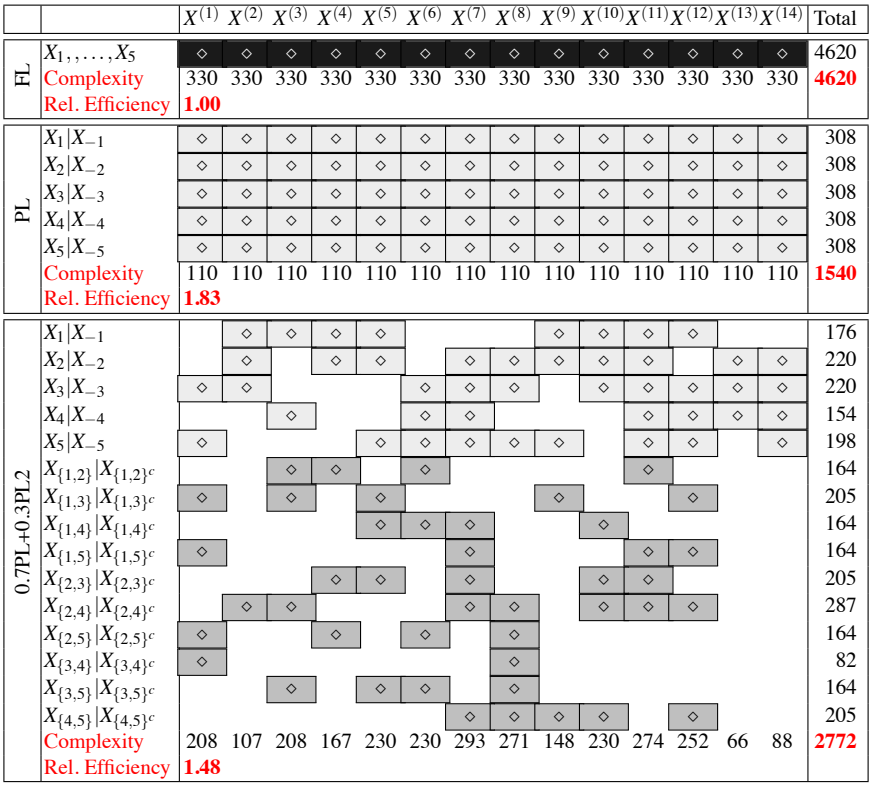


Figure 1: Sample runs of three different SCL policies for 14 examples $X^{(1)},\ldots,X^{(14)}$ drawn from a 5 binary node Boltzmann machine ($\theta^{\text{true}} = (-1,-1,-1,-1,-1,1,1,1,1,1)$). The policies are full likelihood (FL, top), pseudo likelihood (PL, middle), and a stochastic combination of first and second order pseudo likelihood with the first order components selected with probability 0.7 and the second order components with probability 0.3 (bottom).

The sample runs for the policies are illustrated by placing a diamond box in table entries corresponding to selected likelihood objects (rows corresponding to likelihood objects and columns to $X^{(1)},\ldots,X^{(14)}$). The FLOP counts of each likelihood object determines the shade of the diamond boxes while the total FLOP counts per example and per likelihood objects are displayed as table marginals (bottom row and right column for each policy). We also display the total FLOP count and the normalized asymptotic variance (8).

Even in the simple case of 5 nodes, FL is the most complex policy with PL requiring a third of the FL computation. 0.7PL+0.3PL2 is somewhere in between. The situation is reversed for the estimation accuracy-FL achieves the lowest possible normalized asymptotic variance of 1, PL is almost twice that, and 0.7PL+0.3PL2 somewhere in the middle. The SCL framework spans the accuracy-complexity spectrum. Choosing the right λ value obtains an estimator that suits available computational resources and required accuracy.

Definition 3 *A sequence of m-pairs $(A_1,B_1),\ldots,(A_k,B_k)$ is m-pair identifiable, or simply identifiable, of p_θ if the map $\{p_\theta(X_{A_j}|X_{B_j}) : j=1,\ldots,k\} \mapsto p_\theta(X)$ is injective. In other words, there exists only a single collection of conditionals $\{p_\theta(X_{A_j}|X_{B_j}) : j=1,\ldots,k\}$ that does not contradict the joint $p_\theta(X)$.*

Proposition 1 *Let $\Theta \subset \mathbb{R}^r$ be an open set, $p_\theta(x) > 0$ and continuous and smooth in θ, and $(A_1,B_1),\ldots,(A_k,B_k)$ be a sequence of m-pairs for which $\{(A_j,B_j) : \forall j$ such that $\lambda_j > 0\}$ ensures identifiability. Then the sequence of SCL maximizers is strongly consistent, that is,*

$$P\left(\lim_{n\to\infty} \hat{\theta}_n = \theta_0\right) = 1.$$

The above proposition indicates that to guarantee consistency, the sequence of m-pairs needs to satisfy Definition 3. It can be shown that a selection equivalent to the pseudo likelihood function, that is,

$$\mathcal{S} = \{(A_1,B_1),\ldots,(A_m,B_m)\} \quad \text{where} \quad A_i = \{i\}, B_i = \{1,\ldots,m\} \setminus A_i \tag{9}$$

ensures identifiability and consequently the consistency of the MSCLE estimator. Furthermore, every selection of m-pairs that subsumes $\mathcal{S}$ in (9) similarly guarantees identifiability and consistency.

The proposition below establishes the asymptotic normality of the MSCLE $\hat{\theta}_n$. The asymptotic variance enables the comparison of SCL functions with different parameterizations (λ,β).

Proposition 2 *Making the assumptions of Proposition 1 as well as convexity of $\Theta \subset \mathbb{R}^r$ we have the following convergence in distribution*

$$\sqrt{n}(\hat{\theta}_n^{msl} - \theta_0) \rightsquigarrow N(0, \Upsilon\Sigma\Upsilon)$$

where

$$\Upsilon^{-1} = \sum_{j=1}^{k} \beta_j\lambda_j \mathit{Var}_{\theta_0}(\nabla S_{\theta_0}(A_j,B_j)),$$

$$\Sigma = \mathit{Var}_{\theta_0}\left(\sum_{j=1}^{k} \beta_j\lambda_j \nabla S_{\theta_0}(A_j,B_j)\right).$$

The notation $\mathsf{Var}_{\theta_0}(Y)$ represents the covariance matrix of the random vector Y under p_{θ_0} while the notations $\xrightarrow{p}, \rightsquigarrow$ in the proof below denote convergences in probability and in distribution (Ferguson, 1996). ∇ represents the gradient vector with respect to θ.

When θ is a vector the asymptotic variance is a matrix. To facilitate comparison between different estimators we follow the convention of using the determinant, and in some cases the trace, to measure the statistical accuracy. See Chapter 4 of Serfling (1980) for some heuristic arguments for doing so. Figures 1,2,3 provide the asymptotic variance for some SCL estimators and describe how it can be used to gain insight into which estimator to use.

The fact that $\sqrt{n}(\hat{\theta}_n - \theta_0)$ converges in distribution to a Gaussian with zero mean (for the MLE and similarly for SCL estimators as we show above) implies that the estimator's asymptotic behavior, up to $n^{-1/2}$ order, is determined exclusively by the asymptotic variance. That means that the estimator is essentially unbiased up to that order. Higher order statistical analysis (obtained using

Taylor series with more terms) show that the bias decays in the faster rate of n^{-1} (Cox and Snell, 1968). We thus follow the statistical convention of conducting first order asymptotic analysis and concentrate on the estimator's asymptotic variance.

The statistical accuracy of the SCL estimator depends on β (weight parameters) and λ (selection parameter). It is thus desirable to use the results in this section in determining what values of β, λ to use. Directly using the asymptotic variance is not possible in practice as it depends on the unknown quantity θ_0. However, it is possible to estimate the asymptotic variance using the training data. We describe this in Section 7.

5. Robustness of $\hat{\theta}_n^{\mathbf{msl}}$

We observed in our experiments (see Section 8) that the SCL estimator sometimes performed better on a held-out test set than did the maximum likelihood estimator. This phenomenon seems to be in contradiction to the fact that the asymptotic variance of the MLE is lower than that of the SCL maximizer. This is explained by the fact that in some cases the true model generating the data does not lie within the parametric family $\{p_\theta : \theta \in \Theta\}$ under consideration. For example, many graphical models (HMM, CRF, LDA, etc.) make conditional independence assumptions that are often violated in practice. In such cases the SCL estimator acts as a regularizer achieving better test set performance than the non-regularized MLE. We provide below a theoretical account of this phenomenon using the language of m-estimators and statistical robustness. Our notation follows the one in van der Vaart (1998).

We assume that the model generating the data is outside the model family $P(X) \notin \{p_\theta : \theta \in \Theta\}$ and we extend the notation of $m_\theta(X,Z)$ in (7) with,

$$\begin{aligned}
\psi_\theta(X,Z) &\stackrel{\text{def}}{=} \nabla m_\theta(X,Z), \\
\dot{\psi}_\theta(X,Z) &\stackrel{\text{def}}{=} \nabla^2 m_\theta(X,Z), \text{ and} \\
\Psi_n(\theta) &\stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \psi_\theta(X^{(i)},Z^{(i)}),
\end{aligned}$$

noting that $\dot{\psi}_\theta(X,Z)$ is a matrix of second order derivatives.

Proposition 3 below generalizes the consistency result by asserting that $\hat{\theta}_n \to \theta_0$ where θ_0 is the point on $\{p_\theta : \theta \in \Theta\}$ that is closest to the true model P, as defined by

$$\theta_0 = \arg\max_{\theta\in\Theta} M(\theta) \quad \text{where} \quad M(\theta) \stackrel{\text{def}}{=} -\sum_{j=1}^{k} \beta_j \lambda_j D(P(X_{A_j}|X_{B_j})||p_\theta(X_{A_j}|X_{B_j})),$$

or equivalently, θ_0 satisfies

$$\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_{\theta_0}(X,Z) = 0.$$

When the SCL function reverts to the log-likelihood function, θ_0 becomes the KL projection of the true model P onto the parametric family $\{p_\theta : \theta \in \Theta\}$.

Proposition 3 *Assuming the conditions in Proposition 1 as well as* $\sup_{\theta:\|\theta-\theta_0\|\geq\varepsilon} M(\theta) < M(\theta_0)$ *for all* $\varepsilon > 0$ *we have* $\hat{\theta}_n^{msl} \to \theta_0$ *as* $n \to \infty$ *with probability 1.*

The added condition maintains that θ_0 is a well separated maximum point of M. In other words it asserts that only values close to θ_0 may yield a value of M that is close to the maximum $M(\theta_0)$. This condition is satisfied in the case of most exponential family models.

Proposition 4 *Assuming the conditions of Proposition 2 as well as* $E_{P(X)}E_{P(Z)}\|\psi_{\theta_0}(X,Z)\|^2 < \infty$, $E_{P(X)}E_{P(Z)}\psi_{\theta_0}(X)$ *exists and is non-singular,* $|\ddot{\Psi}_{ij}| = |\partial^2\psi_\theta(x)/\partial\theta_i\theta_j| < g(x)$ *for all* i, j *and* θ *in a neighborhood of* θ_0 *for some integrable* g*, we have*

$$\sqrt{n}(\hat{\theta}_n - \theta_0) = -(E_{P(X)}E_{P(Z)}\psi_{\theta_0})^{-1}\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\psi_{\theta_0}(X^{(i)},Z^{(i)}) + o_P(1) \tag{10}$$

or equivalently

$$\hat{\theta}_n = \theta_0 - (E_{P(X)}E_{P(Z)}\psi_{\theta_0})^{-1}\frac{1}{n}\sum_{i=1}^{n}\psi_{\theta_0}(X^{(i)},Z^{(i)}) + o_P\left(\frac{1}{\sqrt{n}}\right). \tag{11}$$

Above, $f_n = o_P(g_n)$ means f_n/g_n converges to 0 with probability 1.

Corollary 1 *Assuming the conditions specified in Proposition 4 we have*

$$\sqrt{n}(\hat{\theta}_n - \theta_0) \rightsquigarrow N(0, (E_{P(X)}E_{P(Z)}\psi_{\theta_0})^{-1}(E_{P(X)}E_{P(Z)}\psi_{\theta_0}\psi_{\theta_0}^{\top})(E_{P(X)}E_{P(Z)}\psi_{\theta_0})^{-1}). \tag{12}$$

Equation (11) means that asymptotically, $\hat{\theta}_n$ behaves as θ_0 plus the average of iid RVs. As mentioned in van der Vaart (1998) this fact may be used to obtain a convenient expression for the asymptotic influence function, which measures the effect of adding a new observation to an existing large data set. Neglecting the remainder in (10) we have

$$\begin{aligned}
I(x,z) &\stackrel{\text{def}}{=} \hat{\theta}_n(X^{(1)},\ldots,X^{(n-1)},x,Z^{(1)},\ldots,Z^{(n-1)},z) - \hat{\theta}_{n-1}(X^{(1)},\ldots,X^{(n-1)},Z^{(1)},\ldots,Z^{(n-1)}) \\
&\approx -(\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_{\theta_0})^{-1}\left(\frac{1}{n}\sum_{i=1}^{n-1}\psi_{\theta_0}(X^{(i)},Z^{(i)}) + \frac{1}{n}\psi_{\theta_0}(w,z) - \frac{1}{n-1}\sum_{i=1}^{n-1}\psi_{\theta_0}(X^{(i)},Z^{(i)})\right) \\
&= -(\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_{\theta_0})^{-1}\frac{1}{n}\psi_{\theta_0}(w,z) + (\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_{\theta_0})^{-1}\frac{1}{n(n-1)}\sum_{i=1}^{n-1}\psi_{\theta_0}(X^{(i)},Z^{(i)}) \\
&= -\frac{1}{n}(\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_{\theta_0})^{-1}\psi_{\theta_0}(w,z) + o_P\left(\frac{1}{n}\right).
\end{aligned} \tag{13}$$

Corollary 1 and Equation (13) measure the statistical behavior of the estimator when the true distribution is outside the model family. In these cases it is possible that a computationally efficient SCL maximizer will result in higher statistical accuracy as well. This "win-win" situation of improving in both accuracy and complexity over the MLE is confirmed by our experiments in Section 8.

We finally note that the above analysis is not limited to misspecified models. For example, the influence function may be used to detect the robustness of $\hat{\theta}_n$ to outliers or rare events (it is desirable to be robust to such occurrences even if the model is not misspecified).

6. Stochastic Composite Likelihood for Markov Random Fields

Markov random fields (MRF) are some of the more popular statistical models for complex high dimensional data. Approaches based on pseudo likelihood and composite likelihood are naturally well-suited in this case due to the cancellation of the normalization term in the probability ratios defining conditional distributions. More specifically, a MRF with respect to a graph $G = (V,E)$, $V = \{1,\ldots,m\}$ with a clique set $\mathcal{C}$ is given by the following exponential family model

$$p_\theta(x) = \exp\left(\sum_{C\in\mathcal{C}} \theta_C f_C(x_C) - \log Z(\theta)\right),$$
$$Z(\theta) = \sum_x \exp\left(\sum_{C\in\mathcal{C}} \theta_c f_C(x_C)\right). \tag{14}$$

The primary bottlenecks in obtaining the maximum likelihood are the computations $\log Z(\theta)$ and $\nabla \log Z(\theta)$. Their computational complexity is exponential in the graph's treewidth and for many cyclic graphs, such as the Ising model or the Boltzmann machine, it is exponential in $|V| = m$.

In contrast, the conditional distributions that form the composite likelihood of (14) are given by (note the cancellation of $Z(\theta)$)

$$p_\theta(x_A|x_B) = \frac{\sum_{x'_{(A\cup B)^c}} \exp\left(\sum_{C\in\mathcal{C}} \theta_C f_C((x_A,x_B,x'_{(A\cup B)^c})_C)\right)}{\sum_{x'_{(A\cup B)^c}} \sum_{x''_A} \exp\left(\sum_{C\in\mathcal{C}} \theta_C f_C((x''_A,x_B,x'_{(A\cup B)^c})_C)\right)}. \tag{15}$$

whose computation is substantially faster. Specifically, The computation of (15) depends on the size of the sets A and $(A\cup B)^c$ and their intersections with the cliques in $\mathcal{C}$. In general, selecting small $|A_j|$ and $B_j = (A_j)^c$ leads to efficient computation of the composite likelihood and its gradient. For example, in the case of $|A_j| = l, |B_j| = m-l$ with $l \ll m$ we have that $k \leq m!/(l!(m-l)!)$ and the complexity of computing the $c\ell(\theta)$ function and its gradient may be shown to require time that is at most exponential in l and polynomial in m.

7. Automatic Selection of β

As Proposition 2 indicates, the weight vector β and selection probabilities λ play an important role in the statistical accuracy of the estimator through its asymptotic variance. The computational complexity, on the other hand, is determined by λ independently of β. Conceptually, we are interested in resolving the accuracy-complexity tradeoff jointly for both β,λ before estimating θ by maximizing the SCL function. However, since the computational complexity depends only on λ we propose the following simplified problem: Select λ based on available computational resources, and then given λ, select the β (and θ) that will achieve optimal statistical accuracy.

Selecting β that minimizes the asymptotic variance is somewhat ambiguous as $\Upsilon\Sigma\Upsilon$ in Proposition 2 is an $r\times r$ positive semidefinite matrix. A common solution is to consider the determinant as a one dimensional measure of the size of the variance matrix,[5] and minimize

$$J(\beta) = \log\det(\Upsilon\Sigma\Upsilon) = \log\det\Sigma + 2\log\det\Upsilon. \tag{16}$$

5. See Chapter 4 of Serfling (1980) for a heuristic discussion motivating this measure.

A major complication with selecting β based on the optimization of (16) is that it depends on the true parameter value θ_0 which is not known at training time. This may be resolved, however, by noting that (16) is composed of covariance matrices under θ_0 which may be estimated using empirical covariances over the training set. To facilitate fast computation of the optimal β we also propose to replace the determinant in (16) with the product of the diagonal elements. Such an approximation is motivated by Hadamard's inequality (which states that for symmetric matrices $\det(M) \leq \prod_i M_{ii}$) and by Geršgorin's circle theorem (see below). This approximation works well in practice as we observe in the experiments section. We also note that the procedure described below involves only simple statistics that may be computed on the fly and does not contribute significant additional computation (nor do they require significant memory).

More specifically, we denote $K^{(ij)} = \mathsf{Cov}_{\theta_0}(\nabla S_{\theta_0}(A_i, B_i), \nabla S_{\theta_0}(A_j, B_j))$ with entries $K^{(ij)}_{st}$, and approximate the log det terms in (16) using

$$\log\det\Upsilon = -\log\det\sum_{j=1}^{k}\beta_j\lambda_j K^{(jj)} \approx -\sum_{l=1}^{r}\log\sum_{j=1}^{k}\beta_j\lambda_j K^{(jj)}_{ll}$$

$$\log\det\Sigma = \log\det\mathsf{Var}_{\theta_0}\left(\sum_{j=1}^{k}\beta_j\lambda_j\nabla S_{\theta_0}(A_j, B_j)\right) = \log\det\sum_{i=1}^{k}\sum_{j=1}^{k}\beta_i\lambda_i\beta_j\lambda_j K^{(ij)}$$

$$\approx \sum_{l=1}^{r}\log\sum_{i=1}^{k}\sum_{j=1}^{k}\beta_i\lambda_i\beta_j\lambda_j K^{(ij)}_{ll}.$$

We denote (assuming A is a $n \times n$ matrix) for $i \in \{1,\ldots,n\}$, $R_i(A) = \sum_{j\neq i}\left|A_{ij}\right|$ and let $D(A_{ii}, R_i(A))$ (D_i where unambiguous) be the closed disc centered at A_{ii} with radius $R_i(A)$. Such a disc is called a Geršgorin disc. The result below states that for matrices that are close to diagonal, the eigenvalues are close to the diagonal elements making our approximation accurate.

Theorem 1 (Geršgorin's circle theorem, for example, Horn and Johnson, 1990) *Every eigenvalue of A lies within at least one of the Geršgorin discs $D(A_{ii}, R_i(A))$. Furthermore, if the union of k discs is disjoint from the union of the remaining $n-k$ discs, then the former union contains exactly k and the latter $n-k$ eigenvalues of A.*

Algorithm 1 solves for θ, β jointly using alternating optimization. The second optimization problem $J(\beta;\cdot)$ is done using the approximation above and may be computed with minimal additional computation. The components of this objective are typically freely available when $sc\ell$ is minimized with Newton-type methods. In practice we found that such an approach leads to a selection of β that is close to optimal, despite loose convergence criteria for the minimization of the $sc\ell$ objective (see Sec. 8.3 and Figures 14, 20 for results).

8. Experiments

We demonstrate the asymptotic properties of $\hat{\theta}^{\mathrm{msl}}_n$ and explore the complexity-accuracy tradeoff for three different models-Boltzmann machine, linear Boltzmann MRF and conditional random fields. In terms of data sets, we consider synthetic data as well as data sets from sentiment prediction and text chunking domains.

In Appendix B we list all figures by subject. The basic road-map is to explore SCL for a theoretical Boltzmann machine and then to explore two data sets using both generative and discriminative models. We also demonstrate the effectiveness of the β heuristic for these experiments.

Algorithm 1 Calculate $\hat{\theta}^{msl}$

Require: $\{X_i\}_{i\in I}$ and $\lambda, \beta^{(0)}$
1: $t \leftarrow 1$
2: **while** $t <$ MAXITS **do**
3: $\quad \theta^{(t)} \leftarrow \arg\min sc\ell(\theta; \{X_i\}_{i\in I}, \lambda, \beta^{(t-1)})$
4: $\quad$ **if** converged **then**
5: $\quad\quad$ **return** θ
6: $\quad$ **end if**
7: $\quad \beta^{(t)} \leftarrow \arg\min J(\beta; \{K^{(ij)}\}_{(i,j)\in J}, \lambda, \theta)$
8: $\quad t \leftarrow t+1$
9: **end while**
10: **return false**

8.1 Toy Example: Boltzmann Machines

We illustrate the improvement in asymptotic variance of the MSCLE associated with adding higher order Boltzmann machine likelihood components with increasingly higher probability. The Boltzmann machine can be parameterized as,

$$p_\theta(x) = \exp\left(\sum_{i<j} \theta_{ij} x_i x_j - \log\psi(\theta)\right), \quad x \in \{0,1\}^m.$$

To be able to accurately compute the asymptotic variance we use $m = 5$ with θ being a $\binom{5}{2}$ dimensional vector with half the components $+1$ and half -1. Since the asymptotic variance of $\hat{\theta}_n^{\text{msl}}$ is a matrix we summarize its size using either its trace or determinant.

Figure 2 displays the asymptotic variance, relative to the minimal variance of the MLE, for the cases of full likelihood (FL), pseudo likelihood ($|A_j| = 1$) PL1, stochastic combination of pseudo likelihood and 2nd order pseudo likelihood ($|A_j| = 2$) components λPL2 $+ (1-\lambda)$PL1, stochastic combination of 2nd order pseudo likelihood and 3rd order pseudo likelihood ($|A_j| = 3$) components λPL3 $+ (1-\lambda)$PL2, and stochastic combination of 3rd order pseudo likelihood and 4th order pseudo likelihood ($|A_j| = 4$) components λPL4 $+ (1-\lambda)$PL3.

The graph demonstrates the computation-accuracy tradeoff as follows: (a) pseudo likelihood is the fastest but also the least accurate, (b) full likelihood is the slowest but the most accurate, (c) adding higher order components reduces the asymptotic variance but also requires more computation, (d) the variance reduces with the increase in the selection probability λ of the higher order component, and (e) adding 4th order components brings the variance very close the lower limit and with each successive improvement becoming smaller and smaller according to a law of diminishing returns.

Figure 3 displays the asymptotic accuracy and complexity for different SCL policies for $m = 9$ binary valued vertices of a Boltzmann machine. We explore three polices in which we denote pseudo likelihood components of size, or order, k. These policies include: $\lambda_1\beta_1$PL1 $+ \lambda_2(1-\beta_1)$PL2, $\lambda_1\beta_1$PL1 $+ \lambda_2(1-\beta_1)$PL3, $\lambda_1\beta_1$PL2 $+ \lambda_2(1-\beta_1)$PL3 (for multiple values of $\lambda_1, \lambda_2, \beta_1$). By taking different linear combinations of various sized pseudo likelihood components, we span a continuous spectrum of accuracy-complexity resolutions. The lower part of the diagram is the boundary of the achievable region (the optimal but unachievable place is the bottom left corner). SCL policies that

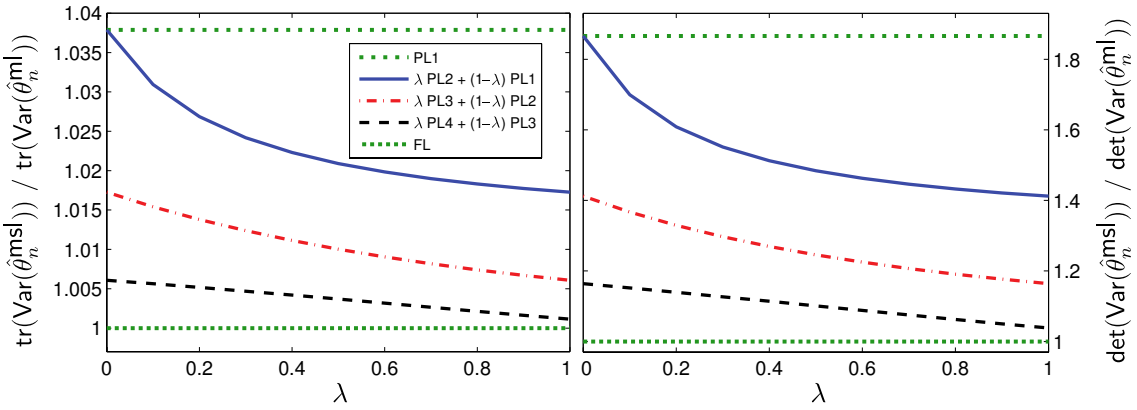


Figure 2: Asymptotic variance matrix, as measured by trace (left) and determinant (right), as a function of the selection probabilities for different stochastic versions of the SCL function.

lie to the right and top of that boundary may be improved by selecting a policy below and to the left of it.

8.2 Local Sentiment Prediction

Our first real world data set experiment involves local sentiment prediction using a conditional MRF model. The data set consisted of 249 movie review documents having an average of 30.5 sentences each with an average of 12.3 words from a 12633 word vocabulary. Each sentence was manually labeled as one of five sentimental designations: very negative, negative, objective, positive, or very positive. As described in Mao and Lebanon (2007) (where more information may be found) we considered the task of predicting the local sentiment flow within these documents using regularized conditional random fields (CRFs) (see Figure 4 for a graphical diagram of the model in the case of four sentences).

As is common practice, we curtail overfitting through a L_2 regularizer, $\exp\{-(2n\sigma^2)^{-1}||\theta||_2^2\}$, which is strong when σ^2 is small and weak when σ^2 is large. We consider σ^2 a hyper-parameter and select it through cross-validation, unless noted otherwise.

Figure 5 shows the contour plots of train and test log-likelihood as a function of the SCL parameters: weight β and selection probability λ. The likelihood components were mixtures of full and pseudo ($|A_j| = 1$) likelihood (rows 1,3) and pseudo and 2nd order pseudo ($|A_j| = 2$) likelihood (rows 2,4). A_j identifies a set of labels corresponding to adjacent sentences over which the probabilistic query is evaluated. Results were averaged over 100 cross validation iterations with 50% train-test split. We used BFGS quasi-Newton method for maximizing the regularized SCL functions. The figure demonstrates how the train log-likelihood increases with increasing the weight and selection probability of full likelihood in rows 1,3 and of 2nd order pseudo likelihood in rows 2,4. This increase in train log-likelihood is also correlated with an increase in computational complexity as higher order likelihood components require more computation. Note however, that the test set behavior in the third and fourth rows shows an improvement in prediction accuracy associated with

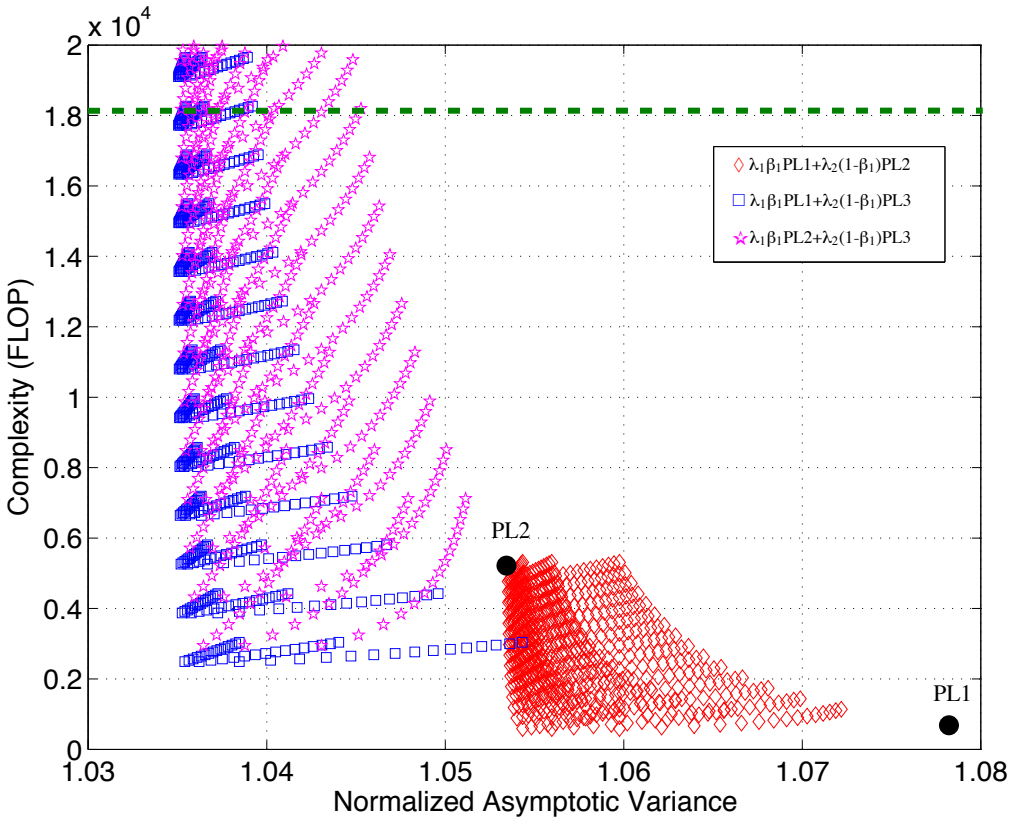


Figure 3: Computation-accuracy diagram for three SCL families:$\lambda_1\beta_1\text{PL1} + \lambda_2(1-\beta_1)\text{PL2}$, $\lambda_1\beta_1\text{PL1} + \lambda_2(1-\beta_1)\text{PL3}$, $\lambda_1\beta_1\text{PL2} + \lambda_2(1-\beta_1)\text{PL3}$ (for multiple values of $\lambda_1, \lambda_2, \beta_1$) for the Boltzmann machine with 9 binary nodes. The pure policies PL1 and PL2 are indicated by black circles and the computational complexity of the full likelihood indicated by a dashed line (corresponding normalized asymptotic variance is 1). The PL3 pure policy is beyond the scale of the diagram. As the graph size increases, the computational cost increases dramatically, in particular for the full likelihood policy and to a lesser extent for the pseudo likelihood policy.

decreasing the influence of full likelihood in favor of pseudo likelihood. The fact that this happens for (relatively) weak regularization, $\sigma^2 = 10$, and indicates that lower order pseudo likelihood has a regularization effect which improves prediction accuracy when the model is not regularized enough. We have encountered this phenomenon in other experiments as well and we will discuss it further in the following subsections.

Figure 6 displays the complexity and negative log-likelihoods (left:train, right:test) of different SCL estimators, sweeping through λ and β, as points in a two dimensional space. The shaded area near the origin is unachievable as no SCL estimator can achieve high accuracy and low computation at the same time. The optimal location in this 2D plane is the curved boundary of the

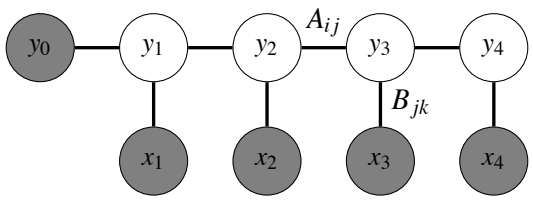


Figure 4: Graphical representation of a four token conditional random field (CRF). A, B are weight matrices and represent state-to-state transitions and state-to-observation outputs. Shading indicates the variable is conditioned upon while no shading indicates the variable is generated by the model.

achievable region with the exact position on that boundary depending on the required solution of the computation-accuracy tradeoff.

8.3 Text Chunking

This experiment consists of using sequential MRFs to divide sentences into "text chunks," that is, syntactically correlated sub-sequences, such as noun and verb phrases. Chunking is a crucial step towards full parsing. For example,[6] the sentence:

> He reckons the current account deficit will narrow to only # 1.8 billion in September.

could be divided as:

> [NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September].

where NP, VP, and PP indicate noun phrase, verb phrase, and prepositional phrase.

We used the publicly available CoNLL-2000 shared task data set. It consists of labeled partitions of a subset of the Wall Street Journal (WSJ) corpus. Our training sets consisted of sampling 100 sentences without replacement from the the CoNLL-2000 training set (211,727 tokens from WSJ Sections 15-18). The test set was the same as the CoNLL-2000 testing partition (47,377 tokens from WSJ Section 20). Each of the possible 21,589 tokens, that is, words, numbers, punctuation, etc., are tagged by one of 11 chunk types and an O label indicating the token is not part of any chunk. Chunk labels are prepended with flags indicating that the token begins (B-) or is inside (I-) the phrase. Figure 7 lists all labels and respective frequencies. In addition to labeled tokens, the data set contains a part-of-speech (POS) column. These tags were automatically generated by the Brill tagger and must be incorporated into any model/feature set accordingly.

In the following, we explore this task using various SCL selection policies on two related, but fundamentally different sequential MRFs: Boltzmann chain MRFs and CRFs.

8.3.1 BOLTZMANN CHAIN MRF

Boltzmann chains are a generative MRF that are closely related to hidden Markov models (HMM). See MacKay (1996) for a discussion on the relationship between Boltzmann chain MRFs and

6. Taken from the CoNLL-2000 shared task, `http://www.cnts.ua.ac.be/conll2000/chunking/`.

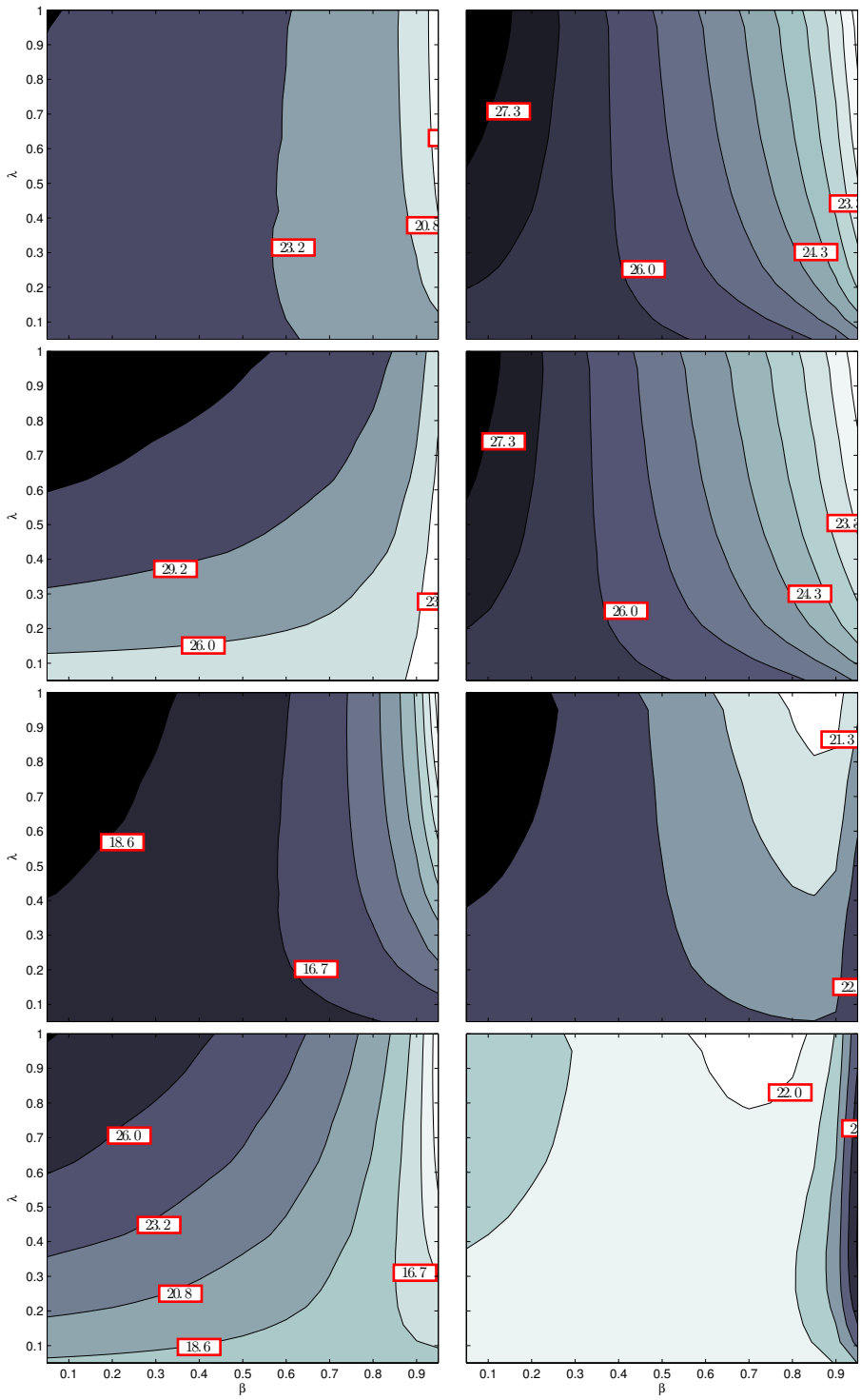


Figure 5: Train (left column) and test (right column) neg. log-likelihood contours for maximum SCL estimators for the CRF model. L_2 regularization, $\exp\{-(2n\sigma^2)^{-1}||\theta||_2^2\}$, parameters are $\sigma^2 = 1$ (rows 1,2) and $\sigma^2 = 10$ (rows 3,4). Rows 1,3 are stochastic mixtures of full (FL) and pseudo (PL1) log-likelihood components while rows 2,4 are PL1 and 2nd order pseudo (PL2). Note that weaker regularization resulted in higher accuracy at lower computation.

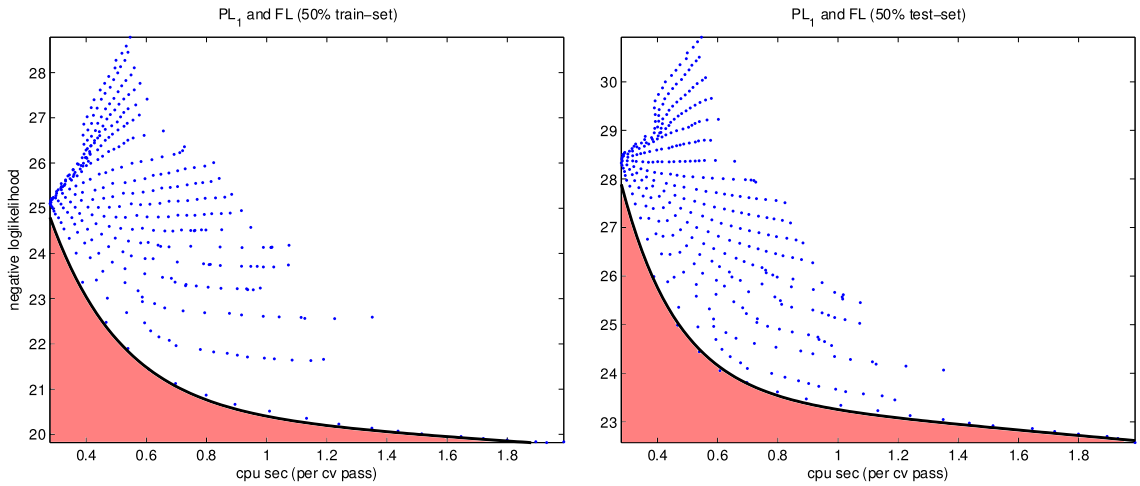


Figure 6: Scatter plot representing complexity and negative log-likelihood (left:train, right:test) of SCL functions for CRFs with L2 regularization parameter $\sigma^2 = 1/2$. The points represent different stochastic combinations of full and pseudo likelihood components. The shaded region represents impossible accuracy/complexity demands. Since the boundary of the obtainable region is empirical, the optimal beta always lies on this boundary. By varying λ, β we are able to smoothly span complexity (wall seconds) and accuracy.

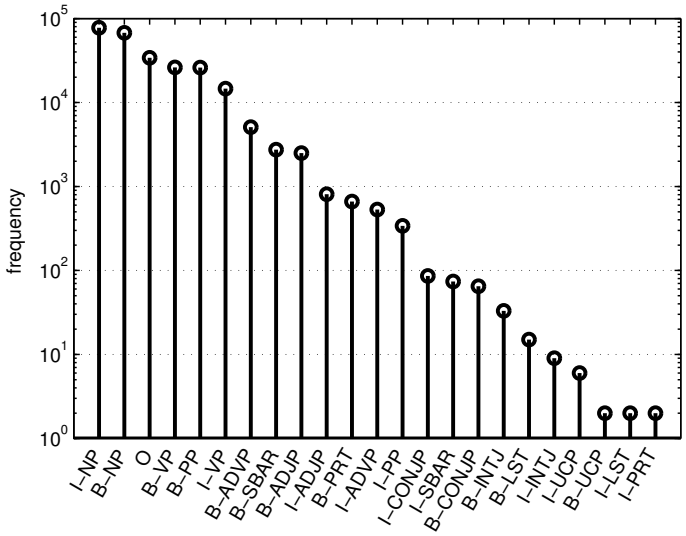


Figure 7: Label counts in CoNLL-2000 data set. Phrases such as noun (NP), verb (VP), and prepositional (PP) are demarcated by a "begin" tag (B-) and an "inside" tag (I-). Non-phrase entities are denoted as "other" (O).

HMMs. We consider SCL components of the form $p(X_2, Y_2|Y_1, Y_3)$, $p(X_2, X_3, Y_2, Y_3|Y_1, Y_4)$ which we refer to as first and second order pseudo likelihood (with higher order components generalizing in a straightforward manner).

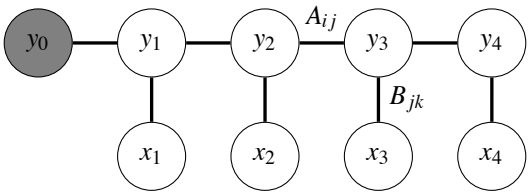


Figure 8: Graphical representation of a four token Boltzmann chain. A, B are weight matrices and represent preference in particular state-to-state transitions and state-to-feature emissions. Only the start state is conditioned upon while all others are generative.

The nature of the Boltzmann chain constrains our feature set to only encode the particular token present at each position, or time index. In doing so we avoid having to model additional dependencies across time steps and dramatically reduce computational complexity. Although SCL is precisely motivated by high treewidth graphs, we wish to include the full likelihood for demonstrative purposes—in practice, this is often not possible. Although POS tags are available we do not include them in these features since the dependence they share on neighboring tokens and other POS tags is unclear. For these reasons our time-sliced feature vector, x_i, has only a single-entry one and cardinality matching the size of the vocabulary (21,589 tokens).

As in Section 8.2, we control overfitting through a L_2 regularizer, $\exp\{-(2n\sigma^2)^{-1}||\theta||_2^2\}$, which is strong when σ^2 is small and weak when σ^2 is large. Here again we choose σ^2 via cross-validation unless otherwise noted. More often though, we show results for several representative σ^2 to demonstrate the roles of λ and β in $\hat{\theta}_n^{msl}$.

Figures 9 and 10 depict train and test negative log-likelihood, that is, perplexity, for the SCL estimator $\hat{\theta}_{100}^{msl}$ with a pseudo/full likelihood selection policy (PL1/FL). As is our convention, weight β and selection probability λ correspond to the higher order component, in this case full likelihood. The lower order pseudo likelihood component is always selected and has weight $1-\beta$. As expected the test set perplexity dominates the train-set perplexity. As was the situation in Sec. 8.2, we note that the lower order component serves to regularize the full likelihood, as evident by the abnormally large σ^2.

We next demonstrate the effect of using a 1st order/2nd order pseudo likelihood selection policy (PL1/PL2). Recall, our notion of pseudo likelihood never entails conditioning on x, although in principle it could. Figures 11 and 12 show how the policy responds to varying both λ and β. Figure 13 depicts the empirical tradeoff between accuracy and complexity. Figure 14 highlights the effectiveness of the β heuristic. See captions for additional comments.

8.3.2 CRFs

Conditional random fields are the discriminative counterpart of Boltzmann chains (cf. Figures 4 and 8). Since x is not jointly modeled with y, we are free to include features with non-independence across time steps without significantly increasing the computational complexity. Here our notion of

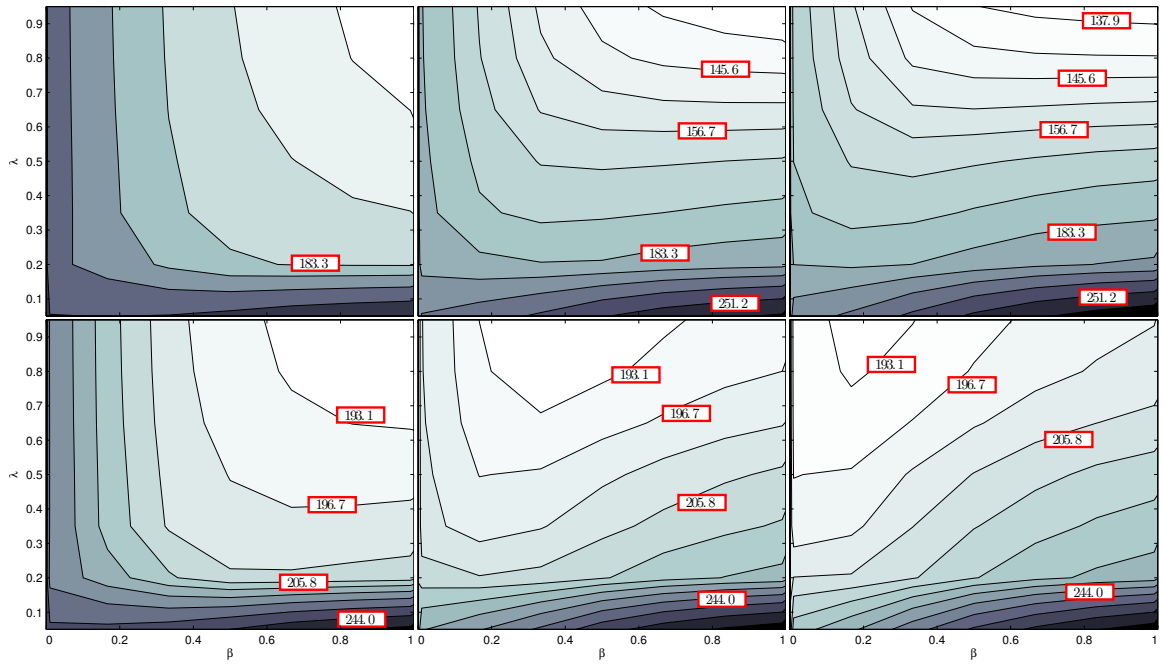


Figure 9: Train set (top) and test set (bottom) negative log-likelihood (perplexity) for the Boltzmann chain MRF with pseudo/full likelihood selection policy (PL1/FL). The x-axis, β, corresponds to relative weight placed on FL and and the y-axis, λ, corresponds to the probability of selecting FL. PL1 is selected with probability 1 and weight $1-\beta$. Contours and labels are fixed across columns. Results averaged over several cross-validation folds, that is, resampling both the train set and the PL1/FL policy. Columns from left to right correspond to weaker regularization, $\sigma^2 = \{500, 2500, 5000\}$. The best achievable test set perplexity is about 190.

Unsurprisingly the test set perplexity dominates the train set perplexity at each σ^2 (column). For a desired level of accuracy (contour) there exists a computationally favorable regularizer. Hence $\hat{\theta}_n^{msl}$ acts as both a regularizer and mechanism for controlling accuracy and complexity.

pseudo likelihood is more traditional, for example, $p(Y_2|Y_1, Y, 3, X_2)$ and $p(Y_2, Y_3|Y_1, Y, 4, X_2, X_3)$ are valid 1st and 2nd order pseudo likelihood components.

We employ a subset of the features outlined in Sha and Pereira (2003) which proved competitive for the CoNLL-2000 shared task. Our feature vector was based on seven feature categories, resulting in a total of 273,571 binary features (i.e., $\sum_i f_i(x_t) = 7$). The feature categories consisted of word unigrams, POS unigrams, word bigrams (forward and backward), and POS bigrams (forward and backward) as well as a stopword indicator (and its complement) as defined by Lewis et al. (2004). The set of possible feature/label pairs is much larger than our set—we use only those features supported by the CoNLL-2000 data set, that is, those which occur at least once. Thus we modeled 297,041 feature/label pairs and 847 transitions for a total of 297,888 parameters. As before, we use

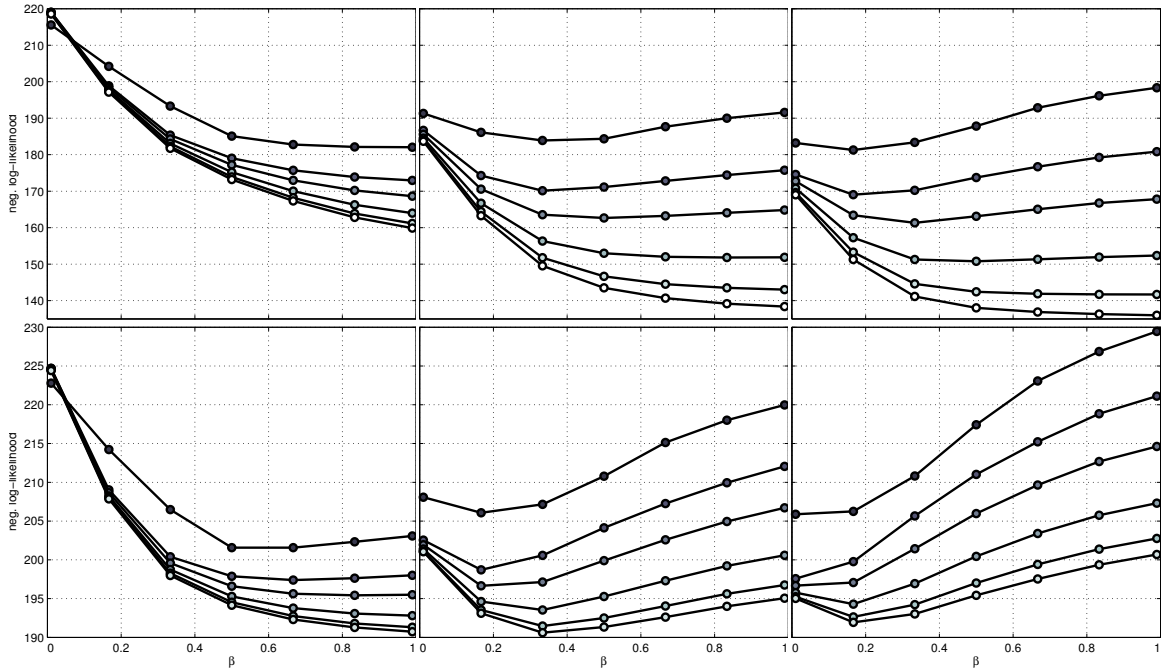


Figure 10: Train set and test set perplexities for the Boltzmann chain MRF with PL1/FL selection policy (see above layout description). The x-axis is again β and the y-axis perplexity. Lighter shading indicates FL is selected with increasing frequency. Note that as the regularizer is weakened the range in perplexity spanned by λ increases and the lower bound decreases. This indicates that the approximating power of $\hat{\theta}_n^{msl}$ increases when unencumbered by the regularizer and highlights its secondary role as a regularizer.

the L_2 regularizer, $\exp\{-(2\sigma^2)^{-1}||\theta||_2^2\}$, which is strong when σ^2 is small and weak when σ^2 is large.

We demonstrate learning on two selection policies: pseudo/full likelihood (Figures 15 and 16) and 1st/2nd order pseudo likelihood (Figures 17 and 18). In both selection polices we note a significant difference from the Boltzmann chain, β has less impact on both train and test perplexity. Intuitively, this seems reasonable as the component likelihood range and variance are constrained by the conditional nature of CRFs. Figure 19 demonstrates the empirical accuracy/complexity tradeoff and Figure 20 depicts the effectiveness of the β heuristic. See captions for further comments.

8.4 Complexity/Regularization Win-Win

It is interesting to contrast the test log-likelihood behavior in the case of mild and stronger L_2 regularization. In the case of weaker or no regularization, the test log-likelihood shows different behavior than the train log-likelihood. Adding a lower order component such as pseudo likelihood acts as a regularizer that prevents overfitting. Thus, in cases that are prone to overfitting reducing higher order likelihood components improves both performance as well as complexity. This represents a win-win situation in contrast to the classical view where the MLE has the lowest variance and adding lower order components reduces complexity but increases the variance.

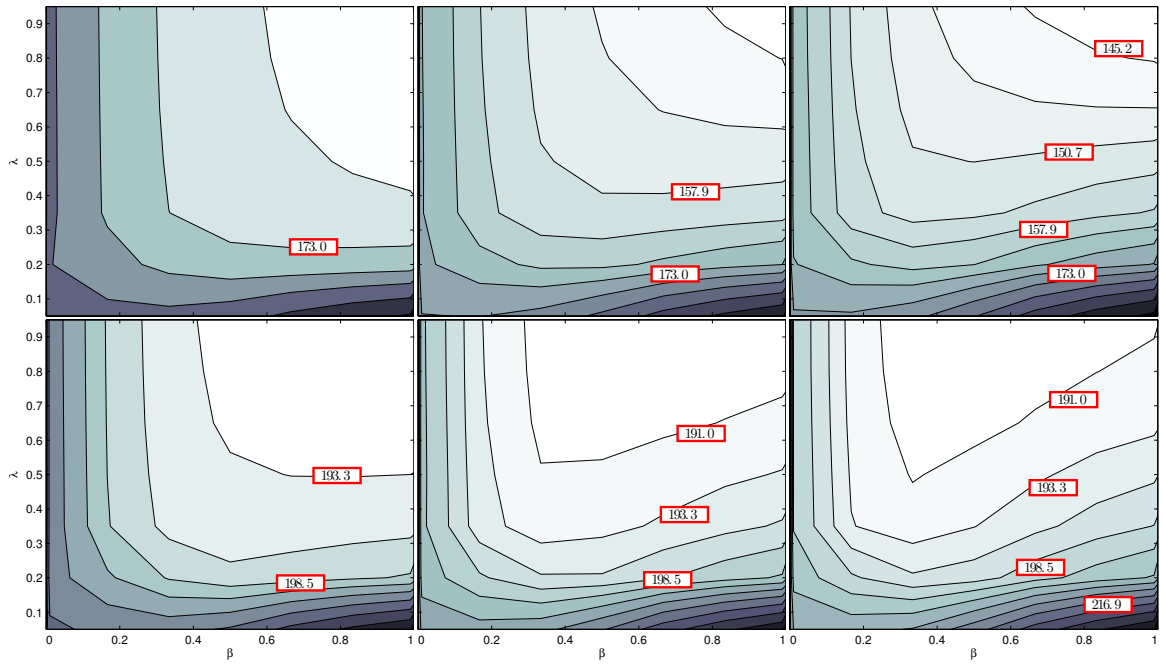


Figure 11: Train set (top) and test set (bottom) perplexity for the Boltzmann chain MRF with 1st/2nd order pseudo likelihood selection policy (PL1/PL2). The x-axis corresponds to PL2 weight and the y-axis the probability of its selection. PL1 is selected with probability 1 and weight $1-\beta$. Columns from left to right correspond to $\sigma^2 = \{5000, 10000, 15000\}$. See Figure 9 for more details. The best achievable test set perplexity is about 189.5.

In comparing these results to PL1/FL, we note that the test set contours exhibit less perplexity for larger areas. In particular, perplexity is lower at smaller λ values, meaning a computational saving over PL1/FL at a given level of accuracy.

In Figure 5 we note this phenomenon when comparing $\sigma^2 = 1$ to $\sigma^2 = 10$ across the selection policies PL1/FL and PL1/PL2. That is, the weaker regularization and more restrictive selection policy, that is, PL1/PL2, is able to achieve comparable test set perplexity.

For the text chunking experiments, we observe a striking win-win when using the Boltzmann chain MRF, Figures 9 and 11. Notice that as regularization is decreased (comparing from left to right), the contours are pulled closer to the x-axis. This means that we are achieving the same perplexity at reduced levels of computational complexity. The CRF however, only exhibits the win-win to a minor extent. We delve deeper into why this is might be the case in the following section.

8.5 λ, σ^2 Interplay

Throughout these experiments we fixed σ^2 and either swept over (λ, β) or used the heuristic to evaluate $(\lambda, \beta(\lambda))$. Motivated by the sometimes weak win-win (cf. Section 8.4) we now consider

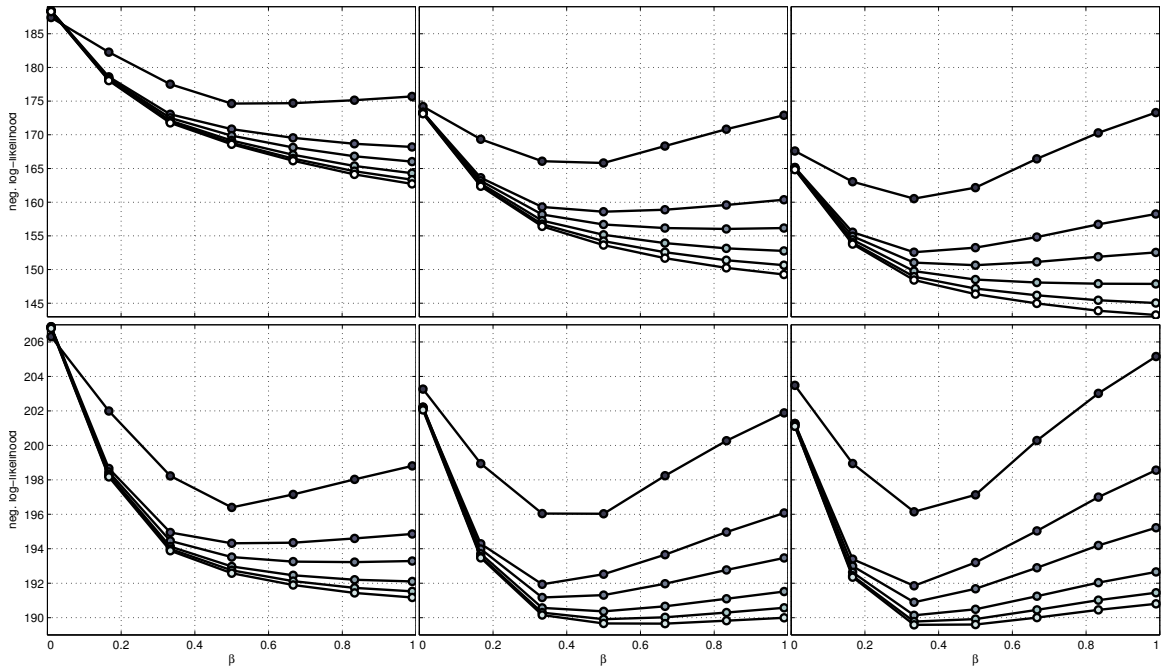


Figure 12: Train (top) and test (bottom) perplexities for the Boltzmann chain MRF with PL1/PL2 selection policy (x-axis:PL2 weight, y-axis:perplexity; see above and previous).

PL1/PL2 outperforms PL1/FL (Fig. 10) test perplexity despite PL1/FL including FL as a special case (i.e., $(\lambda,\beta) = (1,1)$). We speculate that the regularizer's indirect connection to the training samples precludes it from preventing certain types of overfitting. See Sec. 8.4 for more discussion.

how the optimal σ^2 changes as a function of λ. In Figure 21 we used the β heuristic to evaluate train and test perplexity over a (λ,σ^2) grid. We used CRFs and the text chunking task as outlined in Section 8.3.2.

For the PL1/FL policy, we observe that for small enough λ the optimal σ^2, that is, the σ^2 with smallest test perplexity, has considerable range. At some point there are enough samples of the higher-order component to stabilize the choice of regularizer, noting that it is still weaker than the optimal full likelihood regularizer. Conversely, the PL1/PL2 regularizer has an essentially constant optimal regularizer which is relatively much weaker.

As a result, we believe that the lack of win-win for the chunking CRF follows from two effects. In the case of the PL1/FL policy the contour plots are misleading since there is no single σ^2 that performs well across all $\lambda \in [0,1]$. For the PL1/PL2 there is simply little change in regularization necessary across λ.

9. Discussion

The proposed estimator family facilitates computationally efficient estimation in complex graphical models. In particular, different (β,λ) parameterizations of the stochastic composite likelihood en-

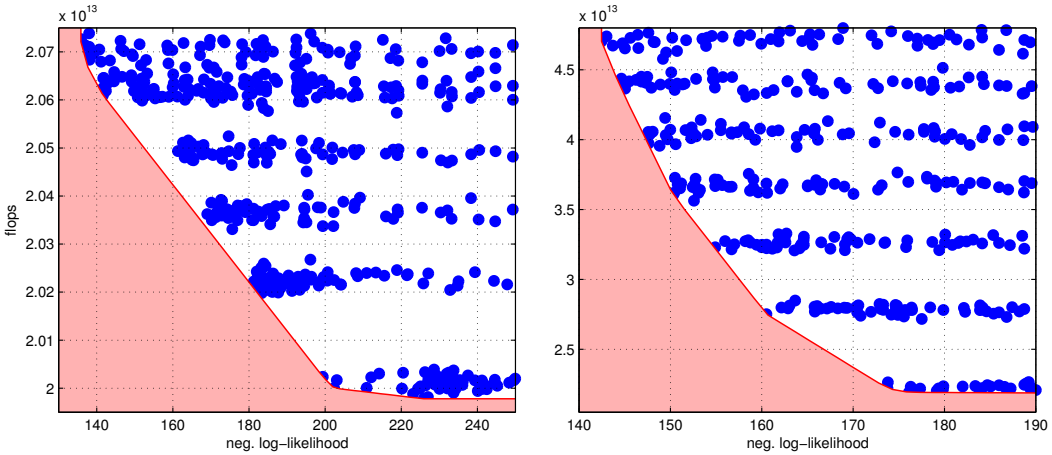


Figure 13: Accuracy and complexity tradeoff for the Boltzmann chain MRF with PL1/FL (left) and PL1/PL2 (right) selection policies. Each point represents the negative log-likelihood (perplexity) and the number of flops required to evaluate the composite likelihood and its gradient under a particular instantiation of the selection policy. The shaded region is the convex hull of the points and represents empirically unobtainable combinations of computational complexity and accuracy. Particularly interesting is the difference between policies and against the discriminative CRF, cf. Figure 19.

ables the resolution of the complexity-accuracy tradeoff in a domain and problem specific manner. The framework is generally suited for Markov random fields, including conditional graphical models and is theoretically motivated. When the model is prone to overfit, stochastically mixing lower order components with higher order ones acts as a regularizer and results in a win-win situation of improving test-set accuracy and reducing computational complexity at the same time.

It is interesting to note that the SCL framework may be generalized to random m-estimators beyond likelihood objects. That is, instead of a fixed m-function we may consider a linear combination of stochastic objects (appearing or not with some probability). Such estimators go beyond traditional m-estimator but may be analyzed using techniques similar to the ones developed in this paper. Although not a random m-estimator, the work of Dillon et al. (2010) borrows SCL concepts to facilitate budgeted semi-supervised learning. This too would benefit from a random m-estimator interpretation and indeed many machine learning tasks may fit nicely into such a framework.

The SCL framework may be useful for a wide variety of intractable graphical models. Besides the examples presented here, it may be particularly suited for large scale models from statistical physics, exponential random graph models, and models from computational biology. A particularly nice feature is that the above computation may be trivially parallelized thus leading to effective computation on large clusters and cloud computing.

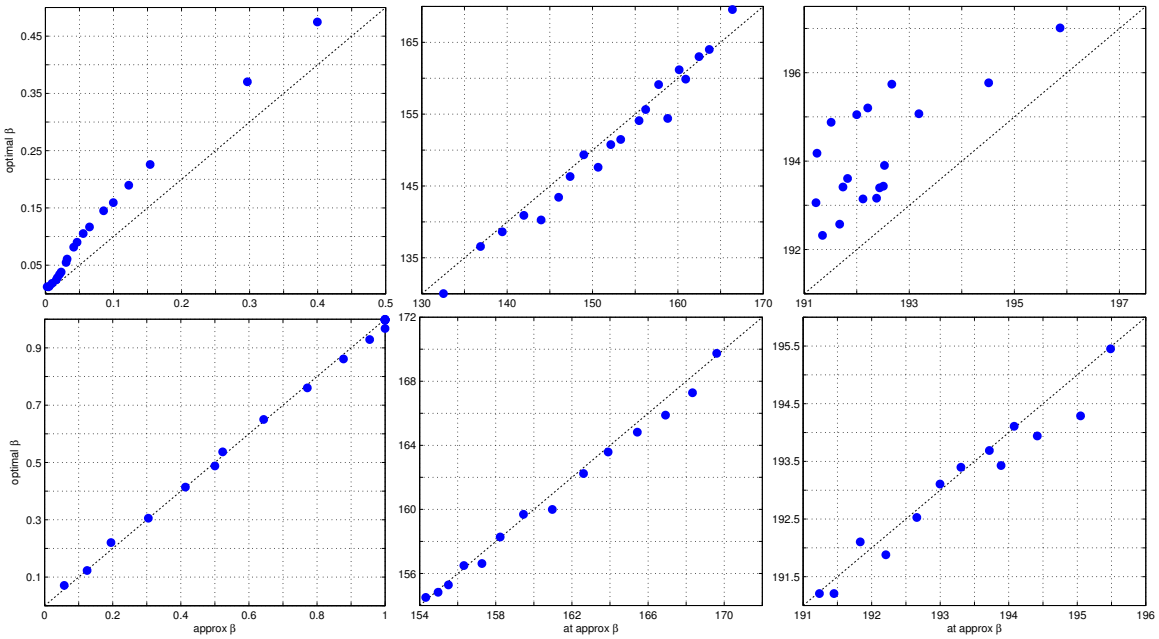


Figure 14: Demonstration of the effectiveness of the β heuristic, that is, using $\hat{\theta}^{msl}$ as a plug-in estimate for θ_0 to periodically re-estimate β during gradient descent. Results are for the Boltzmann chain with PL1/FL (top) and PL1/PL2 (bottom) selection policies. The x-axis is the value at the heuristically found β and the y-axis the value at the optimal β. The optimal β was found by evaluating over a β grid and choosing that with the smallest train set perplexity. The first column depicts the best performing β against the heuristic β. The second and third columns depict the training and testing perplexities (resp.) at the best performing β and heuristically found β. For all three columns, we assess the effectiveness of the heuristic by its nearness to the diagonal (dashed line).

For the PL1/PL2 policy the heuristic closely matched the optimal (all bottom row points are on diagonal). The heuristic out-performed the optimal on the test set and had slightly higher perplexity on the training set. It is a positive result, albeit somewhat surprising, and is attributable to either coarseness in the grid or improved generalization by accounting for variability in $\hat{\theta}^{msl}$.

Acknowledgments

The authors acknowledge useful discussion with P. Liang, M. Welling, P. Smyth, S.-C. Zhu, and C. Sutton. The presentation of the paper was substantially improved following useful feedback from Fernando Pereira and the anonymous referees.

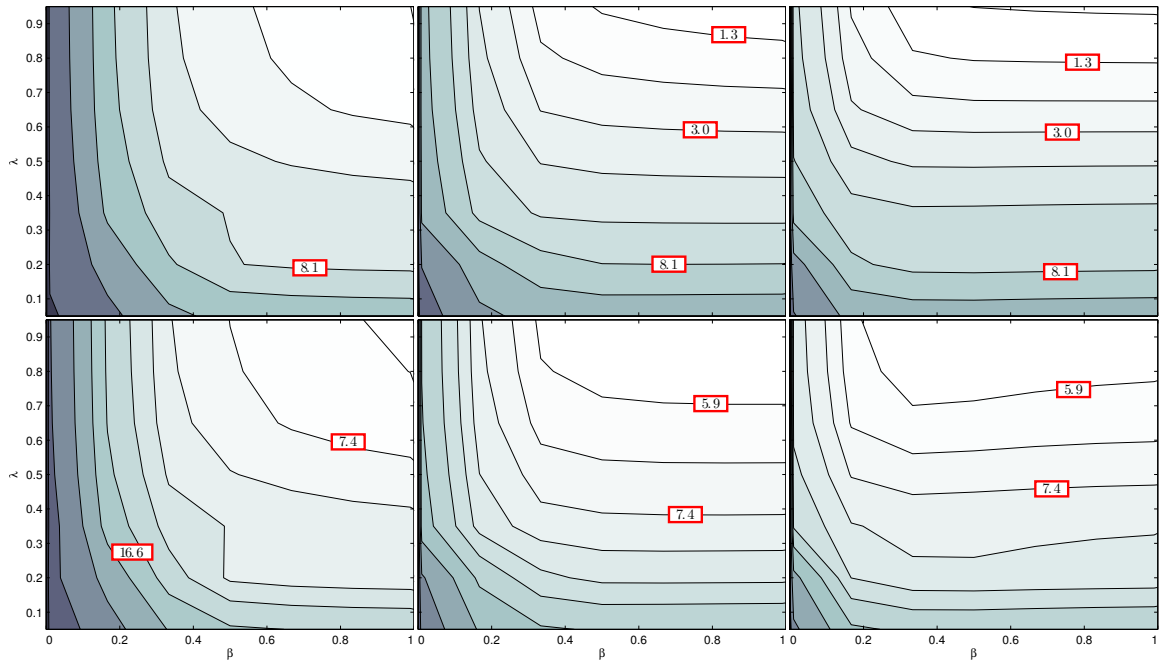


Figure 15: Train set (top) and test set (bottom) perplexity for the CRF with pseudo/full likelihood selection policy (PL1/FL). The x-axis corresponds to FL weight and the y-axis the probability of its selection. PL1 is selected with probability 1 and weight $1-\beta$. Columns from left to right correspond to $\sigma^2 = \{100, 1000, 5000\}$. See Figure 9 for more details. The best achievable test set perplexity is about 5.5.

Although we cannot directly compare CRFs to its generative counterpart, we observe some strikingly different trends. It is immediately clear that the CRF is less sensitive to the relative weighting of components than is the Boltzmann chain. This is partially attributable to a smaller range of the objective—the CRF is already conditional hence the per-component perplexity range is reduced.

Appendix A. Proofs

The proofs below generalize the classical consistency and asymptotic efficiency of the MLE (Ferguson, 1996) and the corresponding results for *m*-estimators (van der Vaart, 1998). They follow similar lines as the proofs in Ferguson (1996) and van der Vaart (1998), with the necessary modifications due to the stochasticity of the SCL function. We assume below that $p_\theta(X) > 0$ and that X is a discrete and finite RV.

The following lemma generalizes Shannon's inequality (Cover and Thomas, 2005) for the KL divergence. We will use it to prove consistency of the SCL estimator.

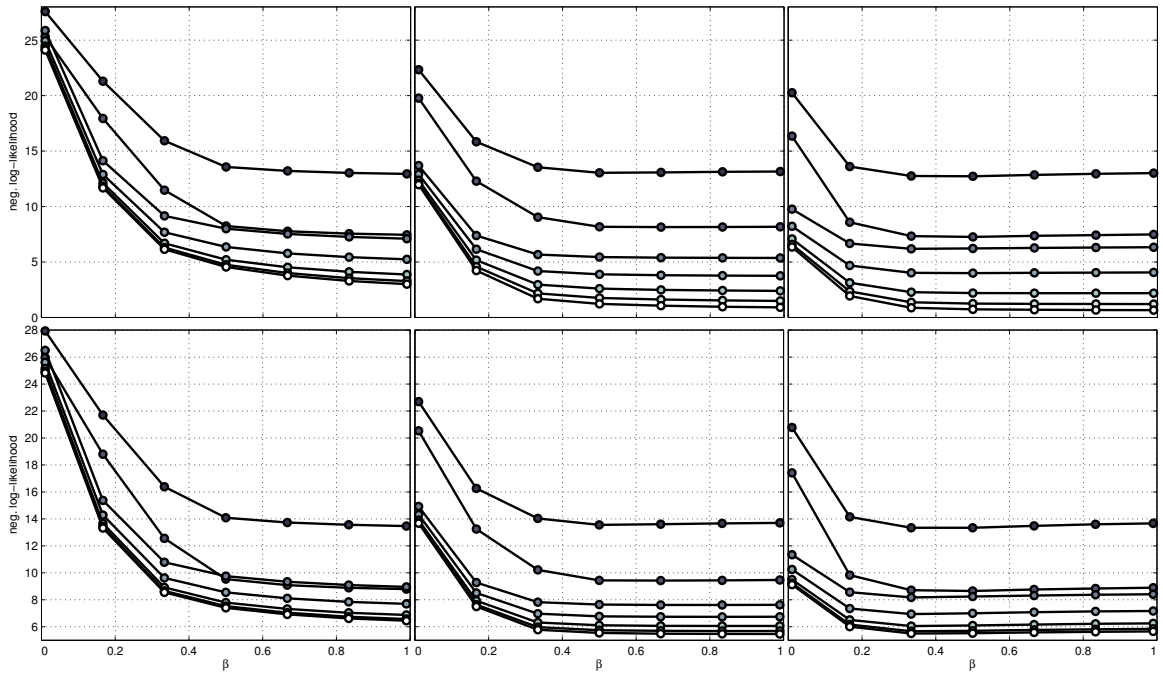


Figure 16: Train (top) and test (bottom) perplexities for a CRF with PL1/FL selection policy (x-axis:FL weight, y-axis:perplexity; see above and Fig. 10).

Perhaps more evidently here than above, we note that the significance of a particular β is less than that of the Boltzmann chain. However, for large enough σ^2, the optimal $\beta \neq 1$. This indicates the dual role of PL1 as a regularizer. Moreover, the left panel calls attention to the interplay between β, λ, and σ^2. See Sec. 8.5 for more discussion.

Lemma 1 *Let* $(A_1,B_1),\ldots,(A_k,B_k)$ *be a sequence of m-pairs that ensures identifiability of* $p_\theta, \theta \in \Theta$ *and* $\alpha_1,\ldots,\alpha_k$ *positive constants. Then*

$$\sum_{j=1}^{k} \alpha_k D(p_\theta(X_{A_j}|X_{B_j}) \,||\, p_{\theta'}(X_{A_j}|X_{B_j})) \geq 0$$

where equality holds iff $\theta = \theta'$.

Proof The inequality follows from applying Jensen's inequality for each conditional KL divergence

$$\begin{aligned} -D(p_\theta(X_{A_j}|X_{B_j}) \,||\, p_{\theta'}(X_{A_j}|X_{B_j})) = \mathsf{E}_{p_\theta} \log \frac{p_{\theta'}(X_{A_j}|X_{B_j})}{p_\theta(X_{A_j}|X_{B_j})} &\leq \log E_{p_\theta} \frac{p_{\theta'}(X_{A_j}|X_{B_j})}{p_\theta(X_{A_j}|X_{B_j})} \\ &= \log 1 = 0. \end{aligned}$$

For equality to hold we need each term to be 0 which follows only if $p_\theta(X_{A_j}|X_{B_j}) \equiv p_{\theta'}(X_{A_j}|X_{B_j})$ for all j which, assuming identifiability, holds iff $\theta = \theta'$. ■

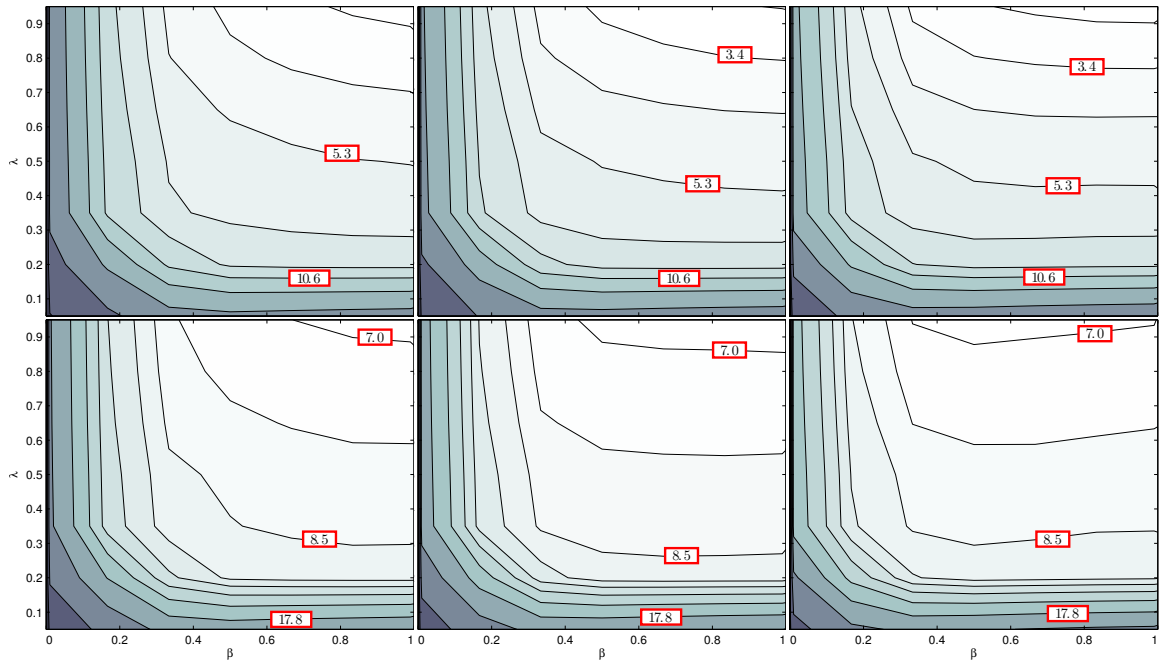


Figure 17: Train set (top) and test set (bottom) perplexity for a CRF with 1st/2nd order pseudo likelihood selection policy (PL1/PL2). The x-axis, β, represents the relative weight placed on PL2 and the y-axis, λ, the probability of selecting PL2. PL1 is selected with probability 1. Columns from left to right correspond to weaker regularization, $\sigma^2 = \{10000, 20000, 40000\}$. See Figure 15 for more details.

Proposition 5 *Let $\Theta \subset \mathbb{R}^r$ be an open set, $p_\theta(x) > 0$ and continuous and smooth in θ, and $(A_1, B_1), \ldots, (A_k, B_k)$ be a sequence of m-pairs for which $\{(A_j, B_j) : \forall j$ such that $\lambda_j > 0\}$ ensures identifiability. Then the sequence of SCL maximizers is strongly consistent, that is,*

$$P\left(\lim_{n\to\infty} \hat{\theta}_n = \theta_0\right) = 1.$$

Proof The SCL function, modified slightly by a linear combination with a term that is constant in θ is

$$sc\ell'(\theta) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} \beta_j \left(Z_{ij} \log p_\theta(X_{A_j}^{(i)} | X_{B_j}^{(i)}) - \lambda_j \log p_{\theta_0}(X_{A_j}^{(i)} | X_{B_j}^{(i)})\right).$$

By the strong law of large numbers, the above expression converges as $n \to \infty$ to its expectation

$$\mu(\theta) = -\sum_{j=1}^{k} \beta_j \lambda_j D(p_{\theta_0}(X_{A_j} | X_{B_j}) \,||\, p_\theta(X_{A_j} | X_{B_j})).$$

If we restrict ourselves to the compact set $S = \{\theta : c_1 \le \|\theta - \theta_0\| \le c_2\}$ then

$$\sup_{\theta \in S} \sup_{Z} \left| \sum_{j=1}^{k} Z_j \beta_j \log p_\theta(X_{A_j} | X_{B_j}) - \lambda_j \beta_j \log p_{\theta_0}(X_{A_j} | X_{B_j}) \right| < K(x) < \infty$$

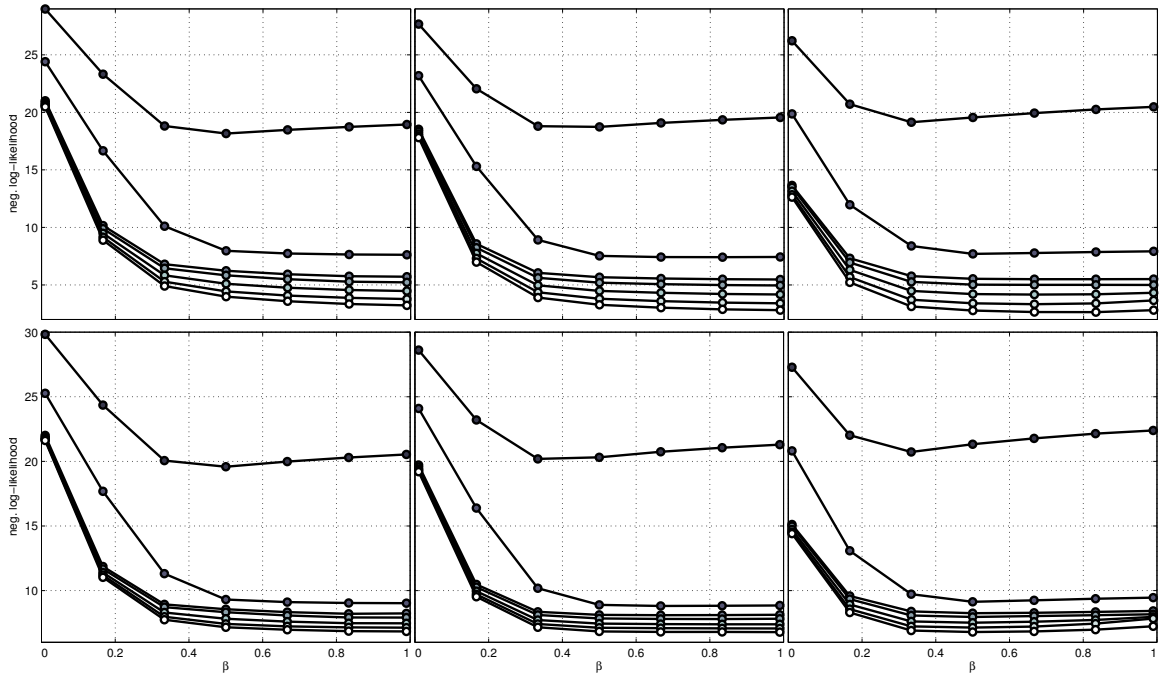


Figure 18: Train (top) and test (bottom) perplexities for a CRF with PL1/PL2 selection policy (x-axis:PL2 weight, y-axis:perplexity; see above and Fig. 10).

Although increasing λ only brings minor improvement to both the training and testing perplexities, it is worth noting that the test perplexity meets that of the PL1/FL. Still though, the overall lack of resolution here suggests that smaller values of λ would better span a range of perplexities and at reduced computational cost.

where $K(x)$ is a function satisfying $\mathsf{E}K(X) < \infty$. As a result, the conditions for the uniform strong law of large numbers (Ferguson, 1996) hold on S leading to

$$P\left\{\lim_{n\to\infty}\sup_{\theta\in S}|scl'(\theta)-\mu(\theta)|=0\right\}=1. \tag{17}$$

By Proposition 1, $\mu(\theta)$ is non-positive and is zero iff $\theta=\theta_0$. Since the function $\mu(\theta)$ is continuous it attains its negative supremum on the compact S: $\sup_{\theta\in S}\mu(\theta)<0$. Combining this fact with (17) we have that there exists N such that for all $n>N$ the SCL maximizers on S achieves strictly negative values of $sc\ell'(\theta)$ with probability 1. However, since $sc\ell'(\theta)$ can be made to achieve values arbitrarily close to zero under $\theta=\theta_0$, we have that $\hat{\theta}_n^{\text{msl}}\notin S$ for $n>N$. Since c_1,c_2 were chosen arbitrarily $\hat{\theta}_n^{\text{msl}}\to\theta_0$ with probability 1. ■

Proposition 6 *Making the assumptions of Proposition 1 as well as convexity of $\Theta\subset\mathbb{R}^r$ we have the following convergence in distribution*

$$\sqrt{n}(\hat{\theta}_n^{msl}-\theta_0)\rightsquigarrow N(0,\Upsilon\Sigma\Upsilon)$$

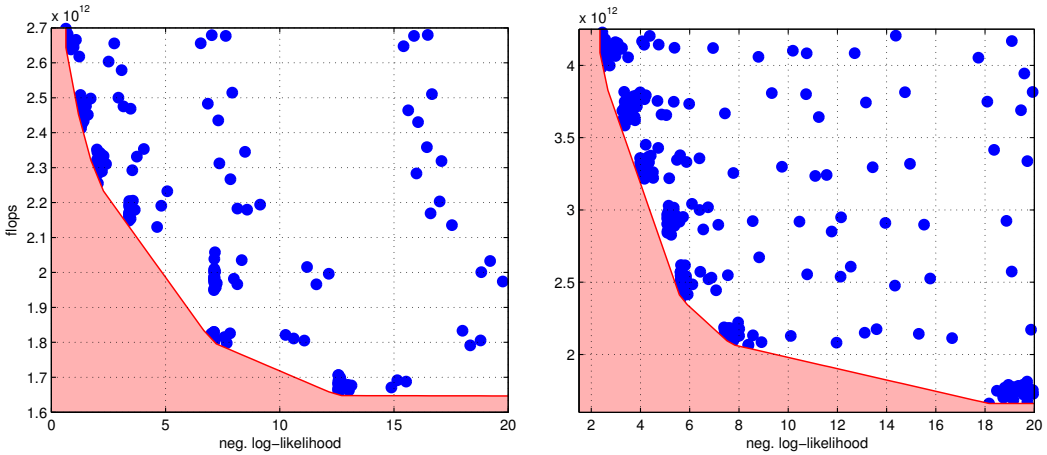


Figure 19: Accuracy and complexity tradeoff for the CRF with PL1/FL (left) and PL1/PL2 (right) selection policies. Each point represents the negative log-likelihood (perplexity) and the number of flops required to evaluate the composite likelihood and its gradient under a particular instance of the selection policy. The shaded region is the convex hull of the points and represents empirically unobtainable combinations of computational complexity and accuracy. σ^2. Particularly interesting is the difference between policies and against the generative Boltzmann chain, cf. Figure 13.

where

$$\Upsilon^{-1} = \sum_{j=1}^{k} \beta_j \lambda_j \mathsf{Var}_{\theta_0}(\nabla S_{\theta_0}(A_j, B_j))$$

$$\Sigma = \mathsf{Var}_{\theta_0}\left(\sum_{j=1}^{k} \beta_j \lambda_j \nabla S_{\theta_0}(A_j, B_j)\right).$$

The notation $\mathsf{Var}_{\theta_0}(Y)$ represents the covariance matrix of the random vector Y under p_{θ_0} while the notations $\xrightarrow{\mathrm{p}}, \rightsquigarrow$ in the proof below denote convergences in probability and in distribution (Ferguson, 1996).

Proof By the mean value theorem and convexity of Θ there exists $\eta \in (0,1)$ for which $\theta' = \theta_0 + \eta(\hat{\theta}_n^{\mathrm{msl}} - \theta_0)$ and

$$\nabla sc\ell_n(\hat{\theta}_n^{\mathrm{msl}}) = \nabla sc\ell_n(\theta_0) + \nabla^2 sc\ell_n(\theta')(\hat{\theta}_n^{\mathrm{msl}} - \theta_0)$$

where $\nabla f(\theta)$ and $\nabla^2 f(\theta)$ are the $r \times 1$ gradient vector and $r \times r$ matrix of second order derivatives of $f(\theta)$. Since $\hat{\theta}_n$ maximizes the SCL, $\nabla sc\ell_n(\hat{\theta}_n^{\mathrm{msl}}) = 0$ and

$$\sqrt{n}(\hat{\theta}_n^{\mathrm{msl}} - \theta_0) = -\sqrt{n}(\nabla^2 sc\ell_n(\theta'))^{-1} \nabla sc\ell_n(\theta_0). \tag{18}$$

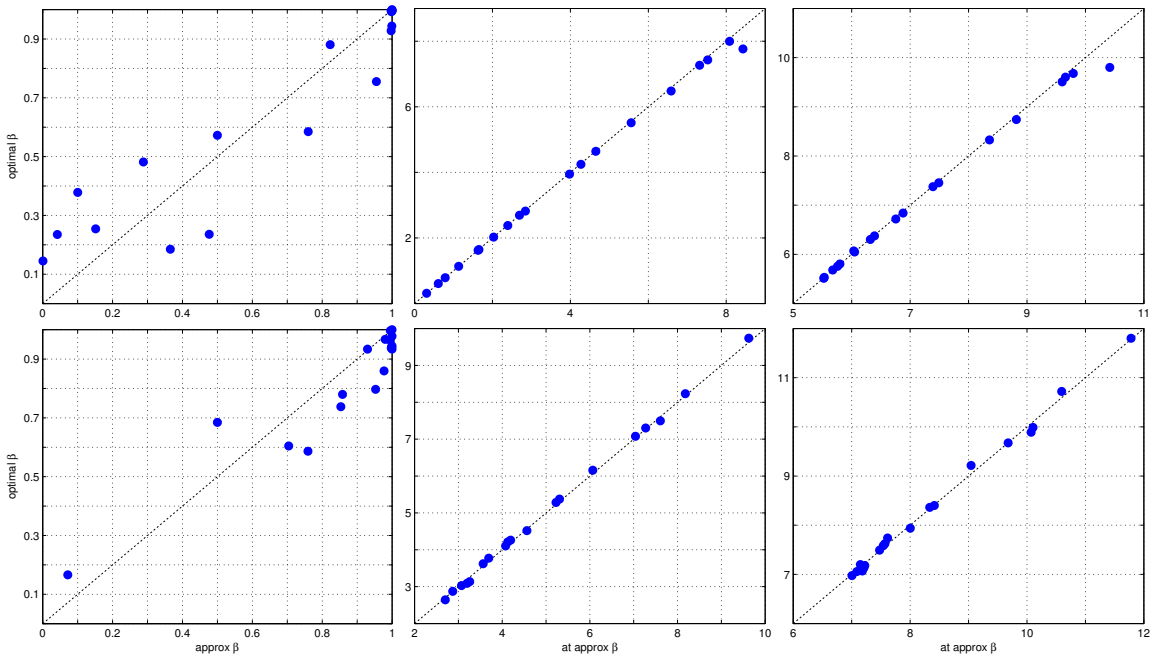


Figure 20: Demonstration of the effectiveness of the β heuristic. Results are for the CRF with PL1/FL (top) and PL1/PL2 (bottom) selection policies. The x-axis is the value at the heuristically found β and the y-axis the value at the optimal β. The first column depicts the best performing β against the heuristic β. The second and third columns depict the training and testing perplexities (resp.) at the best performing β and heuristically found β. For all three columns, we assess the effectiveness of the heuristic by its nearness to the diagonal (dashed line). See Fig. 14 for more details.

The optimal and heuristic β match train and test perplexities for both policies. The actual β value however does not seem to match as well as the Boltzmann chain. However, if we note the flatness of the β grid (cf. Fig. 16 and 18) this result is unsurprising and can be disregarded as an indication of the heuristic's performance.

By Proposition 1 we have $\hat{\theta}_n^{\text{msl}} \xrightarrow{\text{p}} \theta_0$ which implies that $\theta' \xrightarrow{\text{p}} \theta_0$ as well. Furthermore, by the law of large numbers and the fact that if $W_n \xrightarrow{\text{p}} W$ then $g(W_n) \xrightarrow{\text{p}} g(W)$ for continuous g,

$$\begin{aligned}(\nabla^2 sc\ell_n(\theta'))^{-1} &\xrightarrow{\text{p}} (\nabla^2 sc\ell_n(\theta_0))^{-1} \\ &\xrightarrow{\text{p}} \left(\sum_{j=1}^{k} \beta_j \lambda_j \mathsf{E}_{\theta_0} \nabla^2 S_{\theta_0}(A_j, B_j)\right)^{-1} \\ &= -\left(\sum_{j=1}^{k} \beta_j \lambda_j \mathsf{Var}_{\theta_0}(\nabla S_{\theta_0}(A_j, B_j))\right)^{-1}.\end{aligned} \tag{19}$$

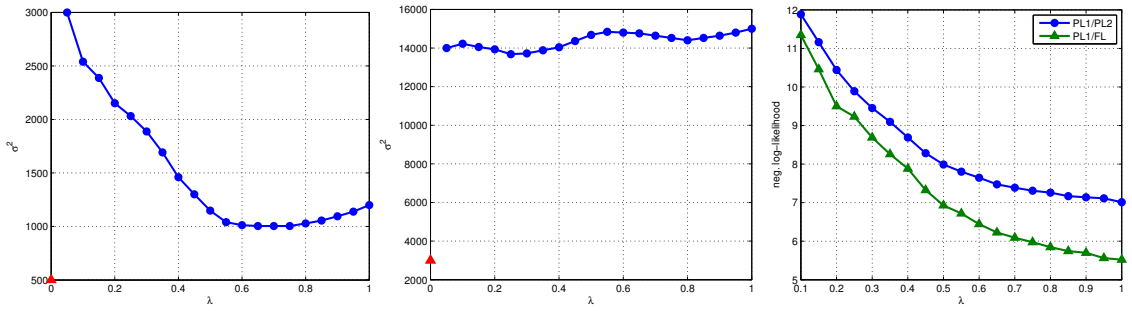


Figure 21: Optimal regularization parameter as a function of $(\lambda, \hat{\beta}(\lambda))$ for PL1/FL (left) and PL1/PL2 (center) CRF selection policies. In the left figure, PL1/FL, λ represents the probability of including FL into the objective. A few FL samples add uncertainty to the objective thus a weaker regularizer is preferable. As more FL samples are incorporated, this effect diminishes but still acts to regularize since the full likelihood (only) best regularization is $\sigma^2 = 500$ (red triangle). The center figure, PL1/PL2, exhibits only a minor change as λ (the probability of incorporating PL2) is increased. It is however, best served by a much weaker regularizer than PL2 alone (red triangle).

The right figure depicts the test-set perplexity as a function of λ using the optimal σ^2 (small λ values were clipped as their performance is quite poor). Note that the perplexity is lowest when both components are always selected ($\lambda = 1$) and that the PL1/FL policy outperforms the PL1/PL2 policy as expected.

For the remaining term in (18) we have

$$\sqrt{n}\nabla sc\ell_n(\theta_0) = \sum_{j=1}^{k} \beta_j \sqrt{n} \frac{1}{n} \sum_{i=1}^{n} W_{ij}$$

where the random vectors $W_{ij} = Z_{ij}\nabla \log p_\theta(X_{A_j}^{(i)} | X_{B_j}^{(i)})$ have expectation 0 and variance matrix $\mathsf{Var}_{\theta_0}(W_{ij}) = \lambda_j \mathsf{Var}_{\theta_0}(\nabla S_{\theta_0}(A_j, B_j))$. By the central limit theorem

$$\sqrt{n}\frac{1}{n}\sum_{i=1}^{n} W_{ij} \rightsquigarrow N\left(0, \lambda_j \mathsf{Var}_{\theta_0}(\nabla S_{\theta_0}(A_j, B_j))\right).$$

The sum $\sqrt{n}\nabla sc\ell_n(\theta_0) = \sum_{j=1}^{k} \beta_j \sqrt{n} \frac{1}{n} \sum_{i=1}^{n} W_{ij}$ is asymptotically Gaussian as well with mean zero since it converges to a sum of Gaussian distributions with mean zero. Since in the general case the random variables $\sqrt{n}\frac{1}{n}\sum_{i=1}^{n} W_{ij}$, $j = 1, \ldots, k$ are correlated, the asymptotic variance matrix of $\sqrt{n}\nabla sc\ell_n(\theta_0)$ needs to account for cross covariance terms leading to

$$\sqrt{n}\nabla sc\ell_n(\theta_0) \rightsquigarrow N\left(0, \mathsf{Var}_{\theta_0}\left(\sum_{j=1}^{k} \beta_j \lambda_j \nabla S_{\theta_0}(A_j, B_j)\right)\right). \tag{20}$$

We finish the proof by combining (18), (19) and (20) using Slutsky's theorem. ■

Recall our notation for the case that the true model $P \notin \{p_\theta : \theta \in \Theta\}$.

$$\psi_\theta(X,Z) \stackrel{\text{def}}{=} \nabla m_\theta(X,Z)$$
$$\dot{\psi}_\theta(X,Z) \stackrel{\text{def}}{=} \nabla^2 m_\theta(X,Z) \quad \text{(matrix of second order derivatives)}$$
$$\Psi_n(\theta) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \psi_\theta(X^{(i)},Z^{(i)}).$$

Proposition 7 *Assuming the conditions in Proposition 1 as well as* $\sup_{\theta:\|\theta-\theta_0\|\geq\varepsilon} M(\theta) < M(\theta_0)$ *for all* $\varepsilon > 0$ *we have* $\hat{\theta}_n^{msl} \to \theta_0$ *as* $n \to \infty$ *with probability 1.*

Proof We assert

$$P\left\{\lim_{n\to\infty} \sup_{\theta\in S} |scl'(\theta) - \mu(\theta)| = 0\right\} = 1. \tag{21}$$

on the compact set $S = \{\theta : c_1 \leq \|\theta - \theta_0\| \leq c_2\}$ as in the proof of Proposition 1. We proceed similarly along the lines of Proposition 1, with the necessary modification due to the fact that the true model is outside the parametric family.

Since the function $\mu(\theta)$ is continuous it attains its negative supremum on the compact S: $\sup_{\theta\in S}\mu(\theta) < \mu(\theta_0) \geq 0$. Combining this fact with (21) we have that there exists N such that for all $n > N$ the SCL maximizers on S achieves strictly negative values of $sc\ell'(\theta)$ with probability 1.

However, since $sc\ell'(\theta)$ can be made to achieve values arbitrarily close to $\mu(\theta_0)$ as $\hat{\theta}_n \to \theta_0$, we have that $\hat{\theta}_n^{\text{msl}} \notin S$ for $n > N$. Since c_1, c_2 were chosen arbitrarily $\hat{\theta}_n^{\text{msl}} \to \theta_0$ with probability 1. ■

Proposition 8 *Assuming the conditions of Proposition 2 as well as* $E_{P(X)}E_{P(Z)}\|\psi_{\theta_0}(X,Z)\|^2 < \infty$, $E_{P(X)}E_{P(Z)}\dot{\psi}_{\theta_0}(X)$ *exists and is non-singular,* $|\ddot{\Psi}_{ij}| = |\partial^2\psi_\theta(x)/\partial\theta_i\theta_j| < g(x)$ *for all* i, j *and* θ *in a neighborhood of* θ_0 *for some integrable* g*, we have*

$$\sqrt{n}(\hat{\theta}_n - \theta_0) = -(E_{P(X)}E_{P(Z)}\dot{\psi}_{\theta_0})^{-1}\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\psi_{\theta_0}(X^{(i)},Z^{(i)}) + o_P(1)$$

or equivalently

$$\hat{\theta}_n = \theta_0 - (E_{P(X)}E_{P(Z)}\dot{\psi}_{\theta_0})^{-1}\frac{1}{n}\sum_{i=1}^{n}\psi_{\theta_0}(X^{(i)},Z^{(i)}) + o_P\left(\frac{1}{\sqrt{n}}\right).$$

Proof By Taylor's theorem there exists a random vector $\tilde{\theta}_n$ on the line segment between θ_0 and $\hat{\theta}_n$ for which

$$0 = \Psi_n(\hat{\theta}_n) = \Psi_n(\theta_0) + \dot{\Psi}_n(\theta_0)(\hat{\theta}_n - \theta_0) + \frac{1}{2}(\hat{\theta}_n - \theta_0)^\top \ddot{\Psi}_n(\tilde{\theta}_n)(\hat{\theta}_n - \theta_0).$$

which we re-arrange as

$$\begin{aligned}\sqrt{n}\dot{\Psi}_n(\theta_0)(\hat{\theta}_n - \theta_0) + \sqrt{n}\frac{1}{2}(\hat{\theta}_n - \theta_0)^\top\ddot{\Psi}_n(\tilde{\theta}_n)(\hat{\theta}_n - \theta_0) &= -\sqrt{n}\Psi_n(\hat{\theta}_n) \\ &= -\sqrt{n}\Psi_n(\theta_0) + o_P(1)\end{aligned} \tag{22}$$

where the second equality follows from the fact that $\hat{\theta}_n \xrightarrow{\text{p}} \theta_0$ and continuous functions preserves converges in probability.

Since $\dot{\Psi}_n(\theta_0)$ converges by the law of large numbers to $\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_\theta(X,Z)$ and $\ddot{\Psi}_n(\tilde{\theta}_n)$ converges to a matrix of bounded values in the neighborhood of θ_0 (for large n), the lhs of (22) is

$$\sqrt{n}\left(\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_\theta(X,Z) + o_P(1) + \frac{1}{2}(\hat{\theta}_n - \theta_0)O_P(1)\right)(\hat{\theta}_n - \theta_0)$$
$$= \sqrt{n}(\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_\theta(X,Z) + o_P(1))(\hat{\theta}_n - \theta_0)$$

since $\hat{\theta}_n - \theta_0 = o_P(1)$ and $o_P(1)O_p(1) = o_P(1)$ (the notation $O_P(1)$ denotes stochastically bounded and it applies to $\ddot{\Psi}_n(\tilde{\theta}_n)$ as described above). Putting it together we have

$$\sqrt{n}(\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_\theta(X,Z) + o_P(1))(\hat{\theta}_n - \theta_0) = -\sqrt{n}\Psi_n(\theta_0) + o_P(1).$$

Since the matrix $\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_\theta(X,Z) + o_P(1)$ converges to a non-singular matrix, multiplying the equation above by its inverse finishes the proof. ■

Corollary 2 *Assuming the conditions specified in Proposition 4 we have*

$$\sqrt{n}(\hat{\theta}_n - \theta_0) \rightsquigarrow N(0, (\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_{\theta_0})^{-1}(\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_{\theta_0}\psi_{\theta_0}^\top)(\mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_{\theta_0})^{-1}).$$

Proof Equation (12) follows from (10) by noticing that due to the central limit theorem $\Psi_n(\theta_0)$ (as it is an average of n iid RVs with expectation 0)

$$\sqrt{n} \cdot \frac{1}{n}\sum_{i=1}^{n} \psi_{\theta_0}(X^{(i)}, Z^{(i)}) \rightsquigarrow N(0, \mathsf{E}_{P(X)}\mathsf{E}_{P(Z)}\psi_{\theta_0}\psi_{\theta_0}^\top).$$

Substituting this in the right hand side of (10) and accounting for the modified variance due to the matrix inverse results in (12). ■

Appendix B. List of Figures

The following lists figures by subject.

B.1 Case Study

- Boltzmann Machines
  - Figure 1 Tabular comparison of policies for computation and accuracy
  - Figure 2 Theoretical analysis of asymptotic variance for trace and determinant
  - Figure 3 Computation/accuracy tradeoff

B.2 Experimental Study

- Labeling sentiment with CRFs
  - Figure 4 CRF graphical model
  - Figure 5 PL1/FL & PL1/PL2 for different σ^2 as a function of $\beta \times \lambda$
  - Figure 6 Computation/accuracy tradeoff with empirical unachievable region

- Chunking CoNLL-2000
  Figure 7 CoNLL-2000 data set label counts

  ... generatively (Boltzmann Chains)
  Figure 8 Boltzmann Chain graphical model
  Figure 13 Computation/accuracy tradeoff with empirical unachievable region
  Figure 9 PL1/FL train & test results as a function of $\beta \times \lambda$
  Figure 10 PL1/FL train & test results as a function of β
  Figure 11 PL1/PL2 train & test results as a function of $\beta \times \lambda$
  Figure 12 PL1/PL2 train & test results as a function of β

  ... discriminatively (CRFs)
  Figure 4 CRF graphical model
  Figure 19 Computation/accuracy tradeoff with empirical unachievable region
  Figure 15 PL1/FL train & test results as a function of $\beta \times \lambda$
  Figure 16 PL1/FL train & test results as a function of β
  Figure 17 PL1/PL2 train & test results as a function of $\beta \times \lambda$
  Figure 18 PL1/PL2 train & test results as a function of β

- β heuristic
  Figure 14 for Boltzmann Chains
  Figure 20 for CRFs
  Figure 21 Optimal regularizing parameter as a function of λ

References

B. Arnold and D. Strauss. Pseudolikelihood estimation: some examples. *Sankhya B*, 53:233–243, 1991.

J. Besag. Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society B*, 36(2):192–236, 1974.

Y. Bishop, S. Fienberg, and P. Holland. *Discrete multivariate analysis: theory and practice*. MIT press, 1975.

Léon Bottou and Olivier Bousquet. Learning using large datasets. In *Mining Massive DataSets for Security*. IOS Press, 2008.

R. Casella and C. Robert. *Monte Carlo Statistical Methods*. Springer Verlag, second edition, 2004.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, second edition, 2005.

D. R. Cox and E. J. Snell. A general definition of residuals (with discussion). *Journal of the Royal Statistical Society B*, 1968.

J. V. Dillon, K. Balasubramanian, and G. Lebanon. Asymptotic analysis of generative semi-supervised learning. In *Proc. of the International Conference on Machine Learning*, 2010.

T. S. Ferguson. *A Course in Large Sample Theory*. Chapman & Hall, 1996.

G. Hinton and T. Sejnowski. Optimal perceptual inference. In *Proc. Computer Vision and Pattern Recognition*, 1983.

N. Hjort and C. Varin. ML, PL, and QL in markov chain models. *Scandinavian Journal of Statistics*, 35(1):64–82, 2008.

R. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

G. Liang and B. Yu. Maximum pseudo likelihood estimation in network tomography. *IEEE Trans. Signal Process*, 51(8):2043–2053, 2003.

P. Liang and M. I. Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *Proc. of the International Conference on Machine Learning*, 2008.

B. G. Lindsay. Composite likelihood methods. *Contemporary Mathematics*, 80:221–239, 1988.

D. J. C. MacKay. Equivalence of linear boltzmann chains and hidden markov models. *Neural Computation*, 8(1):178–181, 1996.

Y. Mao and G. Lebanon. Isotonic conditional random fields and local sentiment flow. In *Advances in Neural Information Processing Systems 19*, pages 961–968, 2007.

R. J. Serfling. *Approximation Theorems of Mathematical Statistics*. John Wiley, 1980.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. *Proceedings of HLT-NAACL*, pages 213–220, 2003.

C. Sutton and A. McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *Proc. of the International Conference on Machine Learning*, 2007.

A. W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

C. Varin and P. Vidoni. A note on composite likelihood inference and model selection. *Biometrika*, 92:519–528, 2005.

D. Vickrey, C. Lin, and D. Koller. Non-local contrastive objectives. In *Proc. of the International Conference on Machine Learning*, 2010.

E. P. Xing, M. I. Jordan, and S. Russell. A generalized mean field algorithm for variational inference in exponential families. In *Proc. of Uncertainty in Artificial Intelligence*, 2003.

S.-C. Zhu and X. Liu. Learning in Gibbsian fields: How accurate and how fast can it be? *IEEE Trans. Pattern Analysis*, 24(7):1001–1006, 2002.

Journal of Machine Learning Research 11 (2010) 2635-2670 Submitted 4/10; Revised 9/10; Published 10/10

Learnability, Stability and Uniform Convergence

Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Givat Ram, Jerusalem 91904, Israel

Ohad Shamir OHADSH@MICROSOFT.COM
Microsoft Research
One Memorial Drive
Cambridge, MA 02142, USA

Nathan Srebro NATI@TTIC.EDU
Karthik Sridharan KARTHIK@TTIC.EDU
Toyota Technological Institute at Chicago
6045 S. Kenwood Ave.
Chicago, IL 60637, USA

Editor: Nicolò Cesa-Bianchi

Abstract

The problem of characterizing learnability is the most basic question of statistical learning theory. A fundamental and long-standing answer, at least for the case of supervised classification and regression, is that learnability is equivalent to uniform convergence of the empirical risk to the population risk, and that if a problem is learnable, it is learnable via empirical risk minimization. In this paper, we consider the General Learning Setting (introduced by Vapnik), which includes most statistical learning problems as special cases. We show that in this setting, there are non-trivial learning problems where uniform convergence does not hold, empirical risk minimization fails, and yet they are learnable using alternative mechanisms. Instead of uniform convergence, we identify stability as the key necessary and sufficient condition for learnability. Moreover, we show that the conditions for learnability in the general setting are significantly more complex than in supervised classification and regression.

Keywords: statistical learning theory, learnability, uniform convergence, stability, stochastic convex optimization

1. Introduction

We consider the General Setting of Learning introduced by Vapnik (1995) where we would like to minimize a population risk functional (stochastic objective)

$$F(\mathbf{h}) = \mathbb{E}_{Z\sim\mathcal{D}}[f(\mathbf{h};Z)] \tag{1}$$

over some hypothesis class $\mathcal{H}$, where the distribution $\mathcal{D}$ of Z is unknown, based on i.i.d. sample $z_1,\ldots,z_m$ drawn from $\mathcal{D}$ (and full knowledge of f and $\mathcal{H}$). This General Setting subsumes supervised classification and regression, certain unsupervised learning problems, density estimation and more. For example, in supervised learning $z=(\mathbf{x},y)$ is an instance-label pair, $\mathbf{h}$ is a predictor, and $f(h;(\mathbf{x},y)) = \text{loss}(h(\mathbf{x}),y)$ is the loss functional. See Section 2 for formal definitions and further examples.

In the context of this general setting, we are concerned with the question of statistical "learnability". That is, when can Equation (1) be minimized to within arbitrary precision based only on a finite sample $z_1,\ldots,z_m$, as $m\to\infty$? We are not concerned here with computational aspects of this problem, that is, whether this approximate minimization can be carried out efficiently, but only whether it is statistically possible to do so based only on the sample $z_1,\ldots,z_m$.

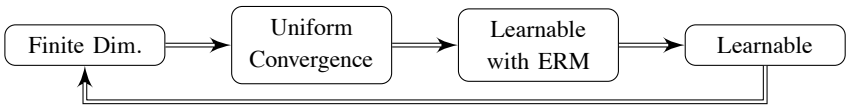
For supervised classification and regression problems, it is well known that a problem is learnable if and only if the empirical risks

$$F_S(\mathbf{h}) = \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{h}, z_i)$$

for all $\mathbf{h} \in \mathcal{H}$ converge uniformly to the population risk (Blumer et al., 1989; Alon et al., 1997). If uniform convergence holds, then the empirical risk minimizer (ERM) is *consistent*, that is, the population risk of the ERM converges to the optimal population risk, and the problem is learnable using the ERM. We therefore have:

- A necessary and sufficient condition for learnability, namely uniform convergence of the empirical risks. Furthermore, this can be shown to be equivalent to a combinatorial condition: having finite VC-dimension in the case of classification, and having finite fat-shattering dimensions in the case of regression.
- A complete understanding of *how* to learn: since learnability is equivalent to learnability by ERM, we can focus our attention solely on empirical risk minimizers.

The situation, for supervised classification and regression, can be depicted as follows:



Other than uniform convergence, certain notions of stability have also been suggested as an explicit condition for learnability. Intuitively, stability notions focus on particular algorithms, or learning rules, and measure their sensitivity to perturbations in the training set. In particular, it is known that stability of the ERM is *sufficient* for learnability. In Mukherjee et al. (2006), it is argued that stability is also a *necessary* for learnability. However, that argument relied on the assumption that uniform convergence is equivalent to learnability. Therefore, stability was shown to characterize learnability only in situations where uniform convergence characterizes learnability anyway.

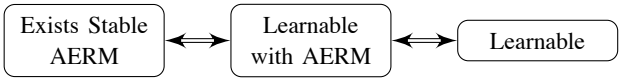
The equivalence of uniform convergence and learnability was formally established only in the supervised classification and regression setting. In the more general setting, the "rightward" implications in the diagram above still hold: finite fat-shattering dimensions, uniform convergence, as well as ERM stability, are indeed sufficient conditions for learnability using the ERM. As for the reverse implication, Vapnik showed that a notion of "non-trivial" or "strict" learnability with the ERM is indeed equivalent to uniform convergence of the empirical risks. This notion was meant to exclude certain "trivial" learning problems, which are learnable without uniform convergence (see Section 3.1). Even in such problems, learnability is still possible by empirical risk minimization. Thus, it would seem that in the General Learning Setting, as in supervised classification and regression, a problem is learnable if and only if it is learnable by empirical risk minimization.

In this paper we show that the situation in the General Learning Setting is actually much more complex. In particular, in Section 4.1 we show an example of a learning problem in the General Learning Setting, which is learnable (using an online algorithm and an online-to-batch conversion), but which is *not* learnable using empirical risk minimization. To the best of our knowledge this is the first example shown of this type.

Furthermore, in Section 4.2 we show a modified example which *is* learnable using empirical risk minimization, but for which the empirical risks of the hypotheses do *not* converge uniformly to their expectations, not even locally for hypotheses very close to the true hypothesis. We argue that unlike the examples discussed in Section 3.1, this example is far from being "trivial". We use this example to discuss how Vapnik's notion of "strict" learnability with the ERM is too strict, and precludes cases which are far from trivial and in which learnability with empirical risk minimization is *not* equivalent to uniform convergence.

Having shown that learnability does not imply learnability with the ERM, and learnability with the ERM does not imply uniform convergence (unlike supervised classification and regression), we proceed in Section 5 to characterize learnability in the General Learning Setting, unveiling stability as a key notion.

In particular, we show that for learnable problems, even when they are not learnable with ERM, they are always learnable with some learning rule which is "asymptotically ERM" and (AERM - see precise definition in Section 2). Moreover, such an AERM must be stable (under a suitable notion of stability). Namely, we have the following characterization of learnability in the General Learning Setting:



Note that this characterization holds even for learnable problems with no uniform convergence. In this sense, stability emerges as a strictly more powerful notion than uniform convergence for characterizing learnability.

Other than this, we also discuss several related results, which above all imply that the conditions for learnability in the General Learning Setting are substantially different and more complex than in supervised classification and regression.

Our results point not to a specific learning rule (such as an ERM), but rather to a class of learning rules (AERM learning rules) as possible candidates for learning. In Section 6, we explore how our results can be strengthened if we allow randomized learning rules. In particular, randomization allows us to pinpoint not a general class of learning rules, but rather a specific (though highly impractical) learning rule, which learns if and only if the problem is learnable.

Throughout most of the paper we discuss learning rates (as a function of the sample size), but do not pay much attention to the confidence at which the learning rule succeeds (i.e., the dependence of the sample size on the allowed probability of failure). This issue is addressed Section 7, and again we show that in the General Learning Setting, things can behave rather differently than in supervised classification and regression.

In summary, this paper opens a door to the complexity of learnability in the General Learning Setting, and provides some understanding of the situation, including highlighting the important role of stability. Many gaps in our understanding remain, and we hope that future progress will close some of these gaps, as well as connect the theoretical insights gained to machine learning as used in practice.

This paper is partially based on the results obtained in Shalev-Shwartz et al. (2009a) and Shalev-Shwartz et al. (2009b).

2. The General Learning Setting: Formal Definition and Notation

In this paper we focus on the General Learning Setting, which was introduced by Vapnik (1995) as a unifying framework for the problem of statistical learning from empirical data.

The General Learning Setting deals with *learning problems*. Formally, a learning problem is specified by a hypothesis class $\mathcal{H}$, an instance set $\mathcal{Z}$ (with a sigma-algebra), and an objective function (e.g., "loss" or "cost") $f : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$. Throughout this paper we assume the function is bounded by some constant B, that is $|f(\mathbf{h};\mathbf{z})| \leq B$ for all $\mathbf{h} \in \mathcal{H}$ and $z \in \mathcal{Z}$.

Given a distribution $\mathcal{D}$ on $\mathcal{Z}$, the quality of each hypothesis $\mathbf{h} \in \mathcal{H}$ is measured by its *risk* $F(\mathbf{h})$, which is defined as $\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}[f(\mathbf{h};\mathbf{z})]$. While $\mathcal{H}$, $\mathcal{Z}$ and $f(\mathbf{h};\mathbf{z})$ are known to the learner, we assume that $\mathcal{D}$ is unknown. Ideally, we would like to pick $\mathbf{h} \in \mathcal{H}$ whose risk is as close as possible to $\inf_{\mathbf{h}\in\mathcal{H}} F(\mathbf{h})$. Since the underlying distribution $\mathcal{D}$ is unknown, we cannot do this directly, but instead need to rely on a finite empirical *training sample* $S = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$. On this sample, we apply a *learning rule* to pick a hypothesis . Formally, a learning rule is a mapping $\mathbf{A} : \cup_{m=1}^{\infty} \mathcal{Z}^m \to \mathcal{H}$ from sequences of instances in $\mathcal{Z}$ to hypotheses. We refer to sequences $S = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ as "sample sets", but it is important to remember that the order and multiplicity of instances may be significant. A learning rule that does not depend on the order of the instances in the training sample is said to be *symmetric*. We will generally consider samples $S \sim \mathcal{D}^m$ of m i.i.d. draws from $\mathcal{D}$.

This framework is sufficiently general to include a large portion of the statistical learning and optimization problems we are aware of, such as:

- **Binary Classification:** Let $\mathcal{Z} = \mathcal{X} \times \{0,1\}$, let $\mathcal{H}$ be a set of functions $\mathbf{h} : \mathcal{X} \mapsto \{0,1\}$, and let $f(\mathbf{h};(\mathbf{x},y)) = \mathbb{1}_{\{\mathbf{h}(\mathbf{x}) \neq y\}}$. Here, $f(\cdot)$ is simply the $0-1$ loss function, measuring whether the binary hypothesis $\mathbf{h}(\cdot)$ misclassified the example $(\mathbf{x},y)$.

- **Regression:** Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are bounded subsets of $\mathbb{R}^n$ and $\mathbb{R}$ respectively, let $\mathcal{H}$ be a set of bounded functions $\mathbf{h} : \mathcal{X}^n \mapsto \mathbb{R}$, and let $f(\mathbf{h};(\mathbf{x},y)) = (\mathbf{h}(\mathbf{x}) - y)^2$. Here, $f(\cdot)$ is simply the squared loss function.

- **Large Margin Classification in a Reproducing Kernel Hilbert Space (RKHS):** Let $\mathcal{Z} = \mathcal{X} \times \{0,1\}$, where $\mathcal{X}$ is a bounded subset of an RKHS, let $\mathcal{H}$ be another bounded subset of the RKHS, and let $f(\mathbf{h};(\mathbf{x},y)) = \max\{0, 1 - y\langle \mathbf{x}, \mathbf{h}\rangle\}$. Here, $f(\cdot)$ is the well known hinge loss function, and our goal is to perform margin-based linear classification in the RKHS.

- **K-Means Clustering in Euclidean Space:** Let $\mathcal{Z} = \mathbb{R}^n$, let $\mathcal{H}$ be all subsets of $\mathbb{R}^n$ of size k, and let $f(\mathbf{h};\mathbf{z}) = \min_{\mathbf{c} \in \mathbf{h}} \|\mathbf{c} - \mathbf{z}\|^2$. Here, each $\mathbf{h}$ represents a set of k centroids, and $f(\cdot)$ measures the Euclidean distance squared between an instance $\mathbf{z}$ and its nearest centroid, according to the hypothesis $\mathbf{h}$.

- **Density Estimation:** Let $\mathcal{Z}$ be a subset of $\mathbb{R}^n$, let $\mathcal{H}$ be a set of bounded probability densities on $\mathcal{Z}$, and let $f(\mathbf{h};\mathbf{z}) = -\log(\mathbf{h}(\mathbf{z}))$. Here, $f(\cdot)$ is simply the negative log-likelihood of an instance $\mathbf{z}$ according to the hypothesis density $\mathbf{h}$. Note that to ensure boundedness of $f(\cdot)$, we need to assume that $\mathbf{h}(\mathbf{z})$ is lower bounded by a positive constant for all $\mathbf{z} \in \mathcal{Z}$.

- **Stochastic Convex Optimization in Hilbert Spaces:** Let $\mathcal{Z}$ be an arbitrary measurable set, let $\mathcal{H}$ be a closed, convex and bounded subset of a Hilbert space, and let $f(\mathbf{h};\mathbf{z})$ be Lipschitz-continuous and convex w.r.t. its first argument. Here, we want to approximately minimize the objective function $\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}[f(\mathbf{h};\mathbf{z})]$, where the distribution over $\mathcal{Z}$ is unknown, based on an empirical sample $\mathbf{z}_1,\ldots,\mathbf{z}_m$.

Our overall goal in this setting is to pick a hypothesis $\mathbf{h} \in \mathcal{H}$ with approximately minimal possible risk, based on a finite sample. Generally, we expect the approximation to get better with the sample size. Learning rules which allow us to choose such hypotheses are said to be *consistent*. Formally, we say a rule $\mathbf{A}$ is consistent with rate $\varepsilon_{\text{cons}}(m)$ under distribution $\mathcal{D}$ if for all m,

$$\mathbb{E}_{S\sim\mathcal{D}^m}[F(\mathbf{A}(S)) - F^*] \leq \varepsilon_{\text{cons}}(m), \tag{2}$$

where we denote $F^* = \inf_{\mathbf{h}\in\mathcal{H}} F(\mathbf{h})$ (here and whenever talking about a "rate" $\varepsilon(m)$, we require it to be monotone decreasing with $\varepsilon_{\text{cons}}(m) \overset{m\to\infty}{\longrightarrow} 0$).

However, since $\mathcal{D}$ is unknown, we cannot choose a learning rule based on $\mathcal{D}$. Instead, we will ask for a stronger requirement, namely that the rule is consistent with rate $\varepsilon_{\text{cons}}(m)$ under *all* distributions $\mathcal{D}$ over $\mathcal{Z}$. This leads to the following central definition:

Definition 1 *A learning problem is* learnable*, if there exist a learning rule* $\mathbf{A}$ *and a monotonically decreasing sequence* $\varepsilon_{\text{cons}}(m)$*, such that* $\varepsilon_{\text{cons}}(m) \overset{m\to\infty}{\longrightarrow} 0$*, and*

$$\forall \mathcal{D}, \quad \mathbb{E}_{S\sim\mathcal{D}^m}[F(\mathbf{A}(S)) - F^*] \leq \varepsilon_{\text{cons}}(m).$$

A learning rule $\mathbf{A}$ *for which this holds is denoted as a* universally consistent *learning rule*.

This definition of learnability, requiring a uniform rate for all distributions, is the relevant notion for studying learnability of a hypothesis class. It is a direct generalization of agnostic PAC-learnability (Kearns et al., 1992) to Vapnik"s General Setting of Learning as studied by Haussler (1992) and others.

A possible approach to learning is to minimize the *empirical risk* $F_S(\mathbf{h})$ over a sample S, defined as

$$F_S(\mathbf{h}) = \frac{1}{m}\sum_{\mathbf{z}\in S} f(\mathbf{h};\mathbf{z}).$$

$\mathcal{Z}$,$\mathbf{z}$	Instance domain and a specific instance.
$\mathcal{H}$,$\mathbf{h}$	Hypothesis class and a specific hypothesis.
$f(\mathbf{h},\mathbf{z})$	Loss of hypothesis $\mathbf{h}$ on instance $\mathbf{z}$
B	$\sup_{\mathbf{h},\mathbf{z}} \lvert f(\mathbf{h};\mathbf{z})\rvert$
$\mathcal{D}$	Underlying distribution on instance domain $\mathcal{Z}$
S	Empirical sample $\mathbf{z}_1,\ldots,\mathbf{z}_m$, sampled i.i.d. from $\mathcal{D}$
m	Size of empirical sample S
$\mathbf{A}(S)$	Learning rule $\mathbf{A}$ applied on empirical sample S
$\varepsilon_{\text{cons}}(m)$	Rate of consistency for a learning rule
$F(\mathbf{h})$	Risk of hypothesis $\mathbf{h}$, $\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}[f(\mathbf{h};\mathbf{z})]$
F^*	$\inf_{\mathbf{h}\in\mathcal{H}} F(\mathbf{h})$
$F_S(\mathbf{h})$	Empirical risk of hypothesis $\mathbf{h}$ on sample S, $\frac{1}{m}\sum_{\mathbf{z}\in S} f(\mathbf{h};\mathbf{z})$
$\hat{\mathbf{h}}_S$	An ERM hypothesis, $F_S(\hat{\mathbf{h}}_S) = \inf_{\mathbf{h}\in\mathcal{H}} F_S(\mathbf{h})$
$\varepsilon_{\text{erm}}(m)$	Rate of AERM for a learning rule
$\varepsilon_{\text{stable}}(m)$	Rate of stability for a learning rule
$\varepsilon_{\text{gen}}(m)$	Rate of generalization for a learning rule

Table 1: Table of Notation

We say that a rule $\mathbf{A}$ is an *ERM (Empirical Risk Minimizer)* if it minimizes the empirical risk

$$F_S(\mathbf{A}(S)) = F_S(\hat{\mathbf{h}}_S) = \inf_{\mathbf{h}\in\mathcal{H}} F_S(\mathbf{h}).$$

where we use $F_S(\hat{\mathbf{h}}_S) = \inf_{\mathbf{h}\in\mathcal{H}} F_S(\mathbf{h})$ to refer to the minimal empirical risk. But since there might be several hypotheses minimizing the empirical risk, $\hat{\mathbf{h}}_S$ does not refer to a specific hypotheses and there might be many rules which are all ERM.

We say that a rule $\mathbf{A}$ is an *AERM (Asymptotic Empirical Risk Minimizer)* with rate $\varepsilon_{\text{erm}}(m)$ under distribution $\mathcal{D}$ if:

$$\mathbb{E}_{S\sim\mathcal{D}^m}\left[F_S(\mathbf{A}(S)) - F_S(\hat{\mathbf{h}}_S)\right] \leq \varepsilon_{\text{erm}}(m)$$

A learning rule is *universally an AERM* with rate $\varepsilon_{\text{erm}}(m)$, if it is an AERM with rate $\varepsilon_{\text{erm}}(m)$ under all distributions $\mathcal{D}$ over $\mathcal{Z}$. A learning rule is an *always AERM* with rate $\varepsilon_{\text{erm}}(m)$, if for *any* sample S of size m, it holds that $F_S(\mathbf{A}(S)) - F_S(\hat{\mathbf{h}}_S) \leq \varepsilon_{\text{erm}}(m)$.

We say a rule $\mathbf{A}$ *generalizes* with rate $\varepsilon_{\text{gen}}(m)$ under distribution $\mathcal{D}$ if for all m,

$$\mathbb{E}_{S\sim\mathcal{D}^m}\left[\lvert F(\mathbf{A}(S)) - F_S(\mathbf{A}(S))\rvert\right] \leq \varepsilon_{\text{gen}}(m).$$

A rule *universally generalizes* with rate $\varepsilon_{\text{gen}}(m)$ if it generalizes with rate $\varepsilon_{\text{gen}}(m)$ under all distributions $\mathcal{D}$ over $\mathcal{Z}$.

We note that other authors sometimes define "consistent", and thus also "learnable" as a combination of our notions of "consistent" and "generalizing".

In the above definitions, we choose to use convergence in expectation, and defined the rates as rates on the expectation. Since the objective f is bounded, convergence in expectation is equivalent to convergence in probability. Furthermore, using Markov's inequality we can translate a rate of the form $\mathbb{E}[\lvert X\rvert] \leq \varepsilon(m)$ to a "low confidence" guarantee $\mathbb{P}[\lvert X\rvert > \varepsilon(m)/\delta] \leq \delta$. See Section 7 for a further discussion on this issue.

3. Background: Characterization of Learnability

Before presenting our results, we begin with a review of the known connections between learnability, stability, and uniform convergence, highlighting the issues which will be of importance later on.

3.1 Learnability and Uniform Convergence

As discussed in the introduction, a central notion for characterizing learnability is uniform convergence. Formally, we say that uniform convergence holds for a learning problem, if the empirical risks of hypotheses in the hypothesis class converges to their population risk uniformly, with a distribution-independent rate:

$$\sup_{\mathcal{D}} \mathbb{E}_{S\sim\mathcal{D}^m}\left[\sup_{\mathbf{h}\in\mathcal{H}} |F(\mathbf{h}) - F_S(\mathbf{h})|\right] \stackrel{m\to\infty}{\longrightarrow} 0.$$

It is straightforward to show that if uniform convergence holds, then a problem can be learned with the ERM learning rule.

For binary classification problems (where $\mathcal{Z} = \mathcal{X} \times \{0,1\}$, each hypothesis is a mapping from $\mathcal{X}$ to $\{0,1\}$, and $f(\mathbf{h};(\mathbf{x},y)) = \mathbb{1}_{\{\mathbf{h}(\mathbf{x})\neq y\}}$), Vapnik and Chervonenkis (1971) showed that the finiteness of a simple combinatorial measure known as the VC-dimension implies uniform convergence. Furthermore, it can be shown that binary classification problems with infinite VC-dimension are not learnable in a distribution-independent sense. This establishes the condition of having finite VC-dimension, and thus also uniform convergence, as a necessary and sufficient condition for learnability.

Such a characterization can also be extended to regression, such as regression with squared loss, where $\mathbf{h}$ is now a real-valued function, and $f(\mathbf{h};(\mathbf{x},y)) = (\mathbf{h}(\mathbf{x}) - y)^2$. The property of having finite fat-shattering dimension at all finite scales now replaces the property of having finite VC-dimension, but the basic equivalence still holds: a problem is learnable if and only if uniform convergence holds (Alon et al., 1997, see also Anthony and Bartlet, 1999, Chapter 19). These results are usually based on clever reductions to binary classification. However, the General Learning Setting that we consider is much more general than classification and regression, and includes setting where a reduction to binary classification is impossible.

To justify the necessity of uniform convergence even in the General Learning Setting, Vapnik attempted to show that in this setting, learnability with the ERM learning rule is equivalent to uniform convergence (Vapnik, 1998). Vapnik noted that this result does not hold, due to "trivial" situations. In particular, consider the case where we take an arbitrary learning problem (with hypothesis class $\mathcal{H}$), and add to $\mathcal{H}$ a single hypothesis $\tilde{\mathbf{h}}$ such that $f(\tilde{\mathbf{h}},\mathbf{z}) < \inf_{\mathbf{h}\in\mathcal{H}} f(\mathbf{h},\mathbf{z})$ for all $\mathbf{z} \in \mathcal{Z}$ (see figure 1 below). This learning problem is now trivially learnable, with the ERM learning rule which always picks $\tilde{\mathbf{h}}$. Note that no assumptions whatsoever are made on $\mathcal{H}$ - in particular, it can be arbitrarily complex, with no uniform convergence or any other particular property. Note also that such a phenomenon is not possible in the binary classification setting, where $f(h;(\mathbf{x},y)) = \mathbb{1}_{\{\mathbf{h}(\mathbf{x})\neq y\}}$, since on any (x,y) we will have hypotheses with $f(h;(\mathbf{x},y)) = f(\tilde{h};(\mathbf{x},y))$ and thus if $\mathcal{H}$ is very complex (has infinite VC dimension) then on every training set there will be many hypotheses with zero empirical error.

To exclude such "trivial" cases, Vapnik introduced a stronger notion of consistency, termed as "strict consistency", which in our notation is defined as

$$\forall c \in \mathbb{R}, \quad \inf_{\mathbf{h}:F(\mathbf{h})\geq c} F_S(\mathbf{h}) \stackrel{m\to\infty}{\longrightarrow} \inf_{\mathbf{h}:F(\mathbf{h})\geq c} F(\mathbf{h}) \ ,$$

where the convergence is in probability. The intuition is that we require the empirical risk of the ERM to converge to the lowest possible risk, even after discarding all the "good" hypotheses whose risk is smaller than some threshold. Vapnik then showed that such strict consistency of the ERM is in fact equivalent to (one-sided) uniform convergence, of the form

$$\sup_{\mathbf{h}\in\mathcal{H}} \left(F(\mathbf{h}) - F_S(\mathbf{h})\right) \stackrel{m\to\infty}{\longrightarrow} 0$$

in probability. Note that this equivalence holds for every distribution separately, and does not rely on universal consistency of the ERM.

These results seem to imply that up to "trivial" situations, a uniform convergence property indeed characterizes learnability, at least using the ERM learning rule. However, as we will see later on, the situation is in fact not that simple.

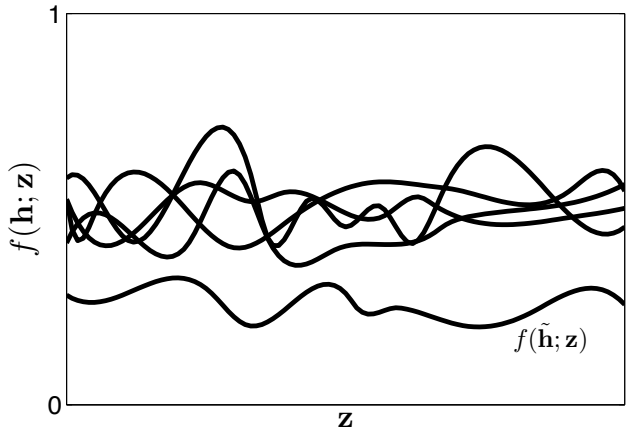


Figure 1: An example of a "trivial" learning situation. Each line represents some $\mathbf{h} \in \mathcal{H}$, and shows the value of $f(\mathbf{h},\mathbf{z})$ for all $\mathbf{z} \in \mathcal{Z}$. The hypothesis $\tilde{\mathbf{h}}$ dominates any other hypothesis (e.g., $f(\tilde{\mathbf{h}};\mathbf{z}) < f(\mathbf{h};\mathbf{z})$ uniformly for all $\mathbf{z}$), and thus the problem is learnable without uniform convergence or any other property of $\mathcal{H}$.

3.2 Learnability and Stability

Instead of focusing on the hypothesis class, and ensuring uniform convergence of the empirical risks of hypothesis in this class, an alternative approach is to directly control the variance of the learning rule. Here, it is not the complexity of the hypothesis class which matters, but rather the way that the learning rule explores this hypothesis class. This alternative approach leads to the notion of stability in learning. It is important to note that stability is a property of a learning rule, not of the hypothesis class.

In the context of modern learning theory,[1] the use of stability can be traced back at least to the work of Rogers and Wagner (1978), which noted that the sensitivity of a learning algorithm with regard to small changes in the sample controls the variance of the leave-one-out estimate. The authors used this observation to obtain generalization bounds (w.r.t. the leave-one-out estimate) for the k-nearest neighbor algorithm. It is interesting to note that a uniform convergence approach for analyzing this algorithm simply cannot work, because the "hypothesis class" in this case has unbounded complexity. These results were later extended to other "local" learning algorithms (see Devroye et al., 1996 and references therein). In addition, practical methods have been developed to introduce stability into learning algorithms, in particular the Bagging technique introduced by Breiman (1996).

Over the last decade, stability was studied as a generic condition for learnability. Kearns and Ron (1999) showed that an algorithm operating on a hypothesis class with finite VC dimension is also stable (under a certain definition of stability). Bousquet and Elisseeff (2002) introduced a strong notion of stability (denoted as *uniform stability*) and showed that it is a sufficient condition for learnability, satisfied by popular learning algorithms such as regularized linear classifiers and regressors in Hilbert spaces (including several variants of SVM). Kutin and Niyogi (2002) introduced several weaker variants of stability, and showed how they are sufficient to obtain generalization bounds for algorithms stable in their sense.

The papers above mainly considered stability as a *sufficient* condition for learnability. A more recent line of work (Rakhlin et al., 2005; Mukherjee et al., 2006) studied stability as a *necessary* condition for learnability. However, the line of argument is specific to settings where uniform convergence holds and is

1. In a more general mathematical context, stability has been around for much longer. The necessity of stability for so-called inverse problems to be well posed was first recognized by Hadamard (1902). The idea of regularization (that is, introducing stability into ill-posed inverse problems) became widely known through the works of Tikhonov (1943) and Phillips (1962). We return to the notion of regularization later on.

necessary for learning. With this assumption, it is possible to show that the ERM algorithm is stable, and thus stability is also a necessary condition for learning. However, as we will see later on in our paper, uniform convergence is in fact not necessary for learning in the General Learning Setting, and stability plays there a key role which has nothing to do with uniform convergence.

Finally, it is important to note that the results cited above make use of many different definitions of stability, which unfortunately are not always comparable. All of them measure stability as the amount of change in the algorithm's output as a function of small changes to the sample on which the algorithm is run. However, "amount of change to the output" and "small changes to the sample" can be defined in many different ways. "Amount of change to the output" can mean change in risk, change in loss with respect to particular examples, or supremum of change in loss over all examples. "Small changes to the sample" usually mean either deleting one example or replacing it with another one (and even here, one can talk about removing/replacing one instance at random, or in some arbitrary manner). Finally, this measure of change can be measured with respect to any arbitrary sample, in expectation over samples drawn from the underlying distribution; or in high probability over samples. For further discussion of this issue, see Appendix A.

4. Gaps Between Learnability, Uniform Convergence and ERM

In this section, we study a special case of the General Learning Setting, where there is a real gap between learnability and uniform convergence, in the sense that there are non-trivial problems where no uniform convergence holds (not even in a local sense), but they are still learnable. Moreover, some of these problems are learnable with an ERM (again, without any uniform convergence), and some are not learnable with an ERM, but rather with a different mechanism. We also discuss why this peculiar behavior does not formally contradict Vapnik's results on the equivalence of strict consistency of the ERM and uniform convergence, as well as the important role that regularization seems to play here, but in a different way than in standard theory.

4.1 Learnability without Uniform Convergence : Stochastic Convex Optimization

A stochastic convex optimization problem is a special case of the General Learning Setting discussed above, with the added constraints that the objective function $f(\mathbf{h};\mathbf{z})$ is Lipschitz-continuous and convex in $\mathbf{h}$ for every $\mathbf{z}$, and that $\mathcal{H}$ is closed, convex and bounded. We will focus here on problems where $\mathcal{H}$ is a subset of a Hilbert space. A special case is the familiar linear prediction setting, where $\mathbf{z} = (\mathbf{x}, y)$ is an instance-label pair, each hypothesis $\mathbf{h}$ belongs to a subset $\mathcal{H}$ of a Hilbert space, and $f(\mathbf{h};\mathbf{x},y) = \ell(\langle \mathbf{h}, \phi(\mathbf{x})\rangle, y)$ for some feature mapping ϕ and a loss function $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$, which is convex w.r.t. its first argument.

The situation in which the stochastic dependence on $\mathbf{h}$ is linear, as in the preceding example, is fairly well understood. When the domain $\mathcal{H}$ and the mapping ϕ are bounded, we have uniform convergence, in the sense that $|F(\mathbf{h}) - F_S(\mathbf{h})|$ is uniformly bounded over all $\mathbf{h} \in \mathcal{H}$ (see Sridharan et al., 2008). This uniform convergence of $F_S(\mathbf{h})$ to $F(\mathbf{h})$ justifies choosing the empirical minimizer $\hat{\mathbf{h}}_S = \arg\min_{\mathbf{h}} F_S(\mathbf{h})$, and guarantees that the expected value of $F(\hat{\mathbf{h}}_S)$ converges to the optimal value $F^* = \inf_{\mathbf{h}} F(\mathbf{h})$.

Even if the dependence on $\mathbf{h}$ is not linear, it is still possible to establish uniform convergence (using covering number arguments) provided that $\mathcal{H}$ is finite dimensional. Unfortunately, when we turn to infinite dimensional hypothesis spaces, uniform convergence might not hold and the problem might not be learnable with empirical minimization. Surprisingly, it turns out that this does not imply that the problem is unlearnable. We will show that using a regularization mechanism, it is possible to devise a learning algorithm for any stochastic convex optimization problem, even when uniform convergence does not hold. This mechanism is fundamentally related to the idea of stability, and will be a good starting point for our more general treatment of stability and learnability in the next section of the paper.

We now turn to discuss our first concrete example. Consider the convex stochastic optimization problem given by

$$f^{(3)}(\mathbf{h};(\mathbf{x},\alpha)) \;=\; \|\alpha * (\mathbf{h}-\mathbf{x})\| \;=\; \sqrt{\sum_i \alpha^2[i](\mathbf{h}[i]-\mathbf{x}[i])^2}\,, \tag{3}$$

where for now we let $\mathcal{H}$ to be the d-dimensional unit sphere $\mathcal{H} = \left\{\mathbf{h} \in \mathbb{R}^d : \|\mathbf{h}\| \leq 1\right\}$, we let $\mathbf{z} = (\mathbf{x}, \alpha)$ with $\alpha \in [0,1]^d$ and $\mathbf{x} \in \mathcal{H}$, and we define $u * v$ to be an element-wise product. We will first consider a sequence of problems, where $d = 2^m$ for any sample size m, and establish that we cannot expect a convergence rate which is independent of the dimensionality d. We then formalize this example in infinite dimensions.

One can think of the problem in Equation (3) as that of finding the "center" of an unknown distribution over $\mathbf{x} \in \mathbb{R}^d$, where we also have stochastic per-coordinate "confidence" measures $\alpha[i]$. We will actually focus on the case where some coordinates are missing, namely that $\alpha[i] = 0$.

Consider the following distribution over $(\mathbf{x}, \alpha)$: $\mathbf{x} = 0$ with probability one, and α is uniform over $\{0,1\}^d$. That is, $\alpha[i]$ are i.i.d. uniform Bernoulli. For a random sample $(\mathbf{x}_1, \alpha_1), \ldots, (\mathbf{x}_m, \alpha_m)$ if $d > 2^m$ then we have that with probability greater than $1 - e^{-1} > 0.63$, there exists a coordinate $j \in 1 \ldots d$ such that all confidence vectors α_i in the sample are zero on the coordinate j, that is $\alpha_i[j] = 0$ for all $i = 1..m$. Let $\mathbf{e}_j \in \mathcal{H}$ be the standard basis vector corresponding to this coordinate. Then

$$F_S^{(3)}(\mathbf{e}_j) \;=\; \frac{1}{m}\sum_{i=1}^{m} \left\|\alpha_i * (\mathbf{e}_j - 0)\right\| \;=\; \frac{1}{m}\sum_{i=1}^{m} |\alpha_i[j]| \;=\; 0,$$

where $F_S^{(3)}(\cdot)$ denotes the empirical risk w.r.t. the function $f^{(3)}(\cdot)$. On the other hand, letting $F^{(3)}(\cdot)$ denote the actual risk w.r.t. $f^{(3)}(\cdot)$, we have

$$F^{(3)}(\mathbf{e}_j) \;=\; \mathbb{E}_{\mathbf{x},\alpha}\left[\left\|\alpha * (\mathbf{e}_j - 0)\right\|\right] \;=\; \mathbb{E}_{\mathbf{x},\alpha}\left[|\alpha[j]|\right] \;=\; 1/2.$$

Therefore, for any m, we can construct a convex Lipschitz-continuous objective in a high enough dimension such that with probability at least 0.63 over the sample, $\sup_{\mathbf{h}} \left|F^{(3)}(\mathbf{h}) - F_S^{(3)}(\mathbf{h})\right| \geq 1/2$. Furthermore, since $f(\cdot;\cdot)$ is non-negative, we have that $\mathbf{e}_j$ is an empirical minimizer, but its expected value $F^{(3)}(\mathbf{e}_j) = 1/2$ is far from the optimal expected value $\min_{\mathbf{h}} F^{(3)}(\mathbf{h}) = F^{(3)}(0) = 0$.

To formalize the example in a sample-size independent way, take $\mathcal{H}$ to be the unit sphere of an infinite-dimensional Hilbert space with orthonormal basis $\mathbf{e}_1, \mathbf{e}_2, \ldots$, where for $\mathbf{v} \in \mathcal{H}$, we refer to its coordinates $\mathbf{v}[j] = \langle \mathbf{v}, \mathbf{e}_j \rangle$ w.r.t this basis. The confidences α are now a mapping of each coordinate to $[0,1]$. That is, an infinite sequence of reals in $[0,1]$. The element-wise product operation $\alpha * \mathbf{v}$ is defined with respect to this basis and the objective function $f^{(3)}(\cdot)$ of Equation (3) is well defined in this infinite-dimensional space.

We again take a distribution over $z = (\mathbf{x}, \alpha)$ where $\mathbf{x} = 0$ and α is an infinite i.i.d. sequence of uniform Bernoulli random variables (that is, a Bernoulli process with each α_i uniform over $\{0,1\}$ and independent of all other α_j). Now, for any finite sample there is almost surely a coordinate j with $\alpha_i[j] = 0$ for all i, and so we a.s. have an empirical minimizer $F_S^{(3)}(\mathbf{e}_j) = 0$ with $F^{(3)}(\mathbf{e}_j) = 1/2 > 0 = F^{(3)}(0)$.

As a result, we see that the empirical values $F_S^{(3)}(\mathbf{h})$ do not converge uniformly to their expectations, and empirical minimization is not guaranteed to solve the problem. Moreover, it is possible to construct a sharper counterexample, in which the *unique* empirical minimizer $\hat{\mathbf{h}}_S$ is far from having optimal expected value. To do so, we augment $f^{(3)}(\cdot)$ by a small term which ensures its empirical minimizer is unique, and far from the origin. Consider:

$$f^{(4)}(\mathbf{h}; (\mathbf{x}, \alpha)) = f^{(3)}(\mathbf{h}; (\mathbf{x}, \alpha)) + \varepsilon \sum_i 2^{-i} (\mathbf{h}[i] - 1)^2 \tag{4}$$

where $\varepsilon = 0.01$. The objective is still convex and $(1+\varepsilon)$-Lipschitz. Furthermore, since the additional term is strictly convex, we have that $f^{(4)}(\mathbf{h}; \mathbf{z})$ is strictly convex w.r.t. $\mathbf{h}$ and so the empirical minimizer is unique.

Consider the same distribution over z: $\mathbf{x} = 0$ while $\alpha[i]$ are i.i.d. uniform zero or one. The empirical minimizer is the minimizer of $F_S^{(4)}(\mathbf{h})$ subject to the constraints $\|\mathbf{h}\| \leq 1$. Identifying the solution to this constrained optimization problem is tricky, but fortunately not necessary. It is enough to show that the optimum of the *unconstrained* optimization problem $h^*_{\mathrm{UC}} = \arg\min F_S^{(4)}(\mathbf{h})$ (without constraining $\mathbf{h} \in \mathcal{H}$) has norm $\|\mathbf{h}^*_{\mathrm{UC}}\| \geq 1$. Notice that in the unconstrained problem, whenever $\alpha_i[j] = 0$ for all $i = 1..n$, only the second term of $f^{(4)}$ depends on $\mathbf{h}[j]$ and we have $\mathbf{h}^*_{\mathrm{UC}}[j] = 1$. Since this happens a.s. for some coordinate j,

we can conclude that the solution to the constrained optimization problem lies on the boundary of $\mathcal{H}$, that is $\left\|\hat{\mathbf{h}}_S\right\| = 1$. But for such a solution we have

$$F^{(4)}(\hat{\mathbf{h}}_S) \geq \mathbb{E}_\alpha\left[\sqrt{\sum_i \alpha[i]\hat{\mathbf{h}}_S^2[i]}\right] \geq \mathbb{E}_\alpha\left[\sum_i \alpha[i]\hat{\mathbf{h}}_S^2[i]\right] = \sum_i \hat{\mathbf{h}}_S^2[i]\mathbb{E}_\alpha\left[\alpha[i]\right] = \frac{1}{2}\left\|\hat{\mathbf{h}}_S\right\|^2 = \frac{1}{2},$$

while $F^* \leq F(0) = \varepsilon$.

In conclusion, no matter how big the sample size is, the unique empirical minimizer $\hat{\mathbf{h}}_S$ of the stochastic convex optimization problem in Equation (4) is a.s. much worse than the population optimum, $F(\hat{\mathbf{h}}_S) \geq \frac{1}{2} > \varepsilon \geq F^*$, and certainly does not converge to it.

4.2 Learnability via Stability

At this point, we have seen an example in the stochastic convex optimization framework where uniform convergence does not hold, and the ERM algorithm fails. Surprisingly, we will now show that such problems are in fact learnable using an alternative mechanism which has nothing to do with uniform convergence.

Given a stochastic convex optimization problem with an objective function $f(\mathbf{h};\mathbf{z})$, consider a *regularized* version of it: instead of minimizing the expected risk $\mathbb{E}_\mathbf{z}\left[f(\mathbf{h};\mathbf{z})\right]$ over $\mathbf{h} \in \mathcal{H}$, we will try to minimize

$$\mathbb{E}_\mathbf{z}\left[f(\mathbf{h};\mathbf{z}) + \frac{\lambda}{2}\left\|\mathbf{h}\right\|^2\right]$$

for some $\lambda > 0$. Notice that this is simply a stochastic convex optimization problem w.r.t. the objective function $f(\mathbf{h};\mathbf{z}) + \frac{\lambda}{2}\left\|\mathbf{h}\right\|^2$. We will show that this regularized problem *is* learnable using the ERM algorithm (namely, by attempting to minimize $\frac{1}{m}\sum_i f(\mathbf{h};\mathbf{z}_i) + \frac{\lambda}{2}\left\|\mathbf{h}\right\|^2$), by showing that the ERM algorithm is *stable*. By taking $\lambda \to 0$ at an appropriate rate as the sample size increases, we are able to solve the original stochastic problem optimization problem, w.r.t. $f(\mathbf{h};\mathbf{z})$.

The key characteristic of the regularized objective function we need is that it is λ-*strongly convex*. Formally, we say that a real function $g(\cdot)$ over a domain $\mathcal{H}$ in a Hilbert space is λ-strongly convex (where $\lambda \geq 0$), if the function $g(\cdot) - \frac{\lambda}{2}\left\|\cdot\right\|^2$ is convex. In this case, it is easy to verify that if $\mathbf{h}$ minimizes g then

$$\forall \mathbf{h}',\ g(\mathbf{h}') - g(\mathbf{h}) \geq \tfrac{\lambda}{2}\left\|\mathbf{h}' - \mathbf{h}\right\|^2 .$$

When $\lambda = 0$, strong convexity corresponds to standard convexity. In particular, it is immediate from the definition that $f(\mathbf{h};\mathbf{z}) + \frac{\lambda}{2}\left\|\mathbf{h}\right\|^2$ is λ-strongly convex w.r.t. $\mathbf{h}$ (assuming $f(\mathbf{h};\mathbf{z})$ is convex).

The arguments above are formalized in the following two theorems:

Theorem 2 *Consider a stochastic convex optimization problem such that $f(\mathbf{h};\mathbf{z})$ is λ-strongly convex and L-Lipschitz with respect to $\mathbf{h} \in \mathcal{H}$. Let $\mathbf{z}_1,\ldots,\mathbf{z}_m$ be an i.i.d. sample and let $\hat{\mathbf{h}}_S$ be the empirical minimizer. Then, with probability at least $1-\delta$ over the sample we have*

$$F(\hat{\mathbf{h}}_S) - F^* \ \leq\ \frac{4L^2}{\delta\lambda m} .$$

Theorem 3 *Let $f : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$ be such that $\mathcal{H}$ is bounded by B and $f(\mathbf{h},\mathbf{z})$ is convex and L-Lipschitz with respect to $\mathbf{h}$. Let $\mathbf{z}_1,\ldots,\mathbf{z}_m$ be an i.i.d. sample and let $\hat{\mathbf{h}}_\lambda$ be the minimizer of*

$$\hat{\mathbf{h}}_\lambda = \min_{\mathbf{h}\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^m f(\mathbf{h},\mathbf{z}_i) + \tfrac{\lambda}{2}\left\|\mathbf{h}\right\|^2\right) \tag{5}$$

with $\lambda = \sqrt{\frac{16L^2}{\delta B^2 m}}$. Then, with probability at least $1-\delta$ we have

$$F(\hat{\mathbf{h}}_\lambda) - F^* \ \leq\ 4\sqrt{\frac{L^2B^2}{\delta m}}\left(1 + \frac{8}{\delta m}\right) .$$

Proof [Proof of Theorem 2] To prove the theorem, we use a stability argument. Denote

$$F_S^{(i)}(\mathbf{h}) = \frac{1}{m}\left(f(\mathbf{h},\mathbf{z}_i') + \sum_{j\neq i} f(\mathbf{h},\mathbf{z}_j)\right).$$

the empirical average with $\mathbf{z}_i$ replaced by an independently and identically drawn $\mathbf{z}_i'$, and consider its minimizer:

$$\hat{\mathbf{h}}_S^{(i)} = \arg\min_{\mathbf{h}\in\mathcal{H}} F_S^{(i)}(\mathbf{h}).$$

We first use strong convexity and Lipschitz-continuity to establish that empirical minimization is stable in the following sense:

$$\forall \mathbf{z}\in\mathbb{Z},\quad \left|f(\hat{\mathbf{h}}_S,\mathbf{z}) - f(\hat{\mathbf{h}}_S^{(i)},\mathbf{z})\right| \leq \tfrac{4L^2}{\lambda m}\,. \tag{6}$$

We have that

$$\begin{aligned}
&F_S(\hat{\mathbf{h}}_S^{(i)}) - F_S(\hat{\mathbf{h}}_S)\\
&= \frac{f(\hat{\mathbf{h}}_S^{(i)},\mathbf{z}_i) - f(\hat{\mathbf{h}}_S,\mathbf{z}_i)}{m} + \frac{\sum_{j\neq i}\left(f(\hat{\mathbf{h}}_S^{(i)},\mathbf{z}_j) - f(\hat{\mathbf{h}}_S,\mathbf{z}_j)\right)}{m}\\
&= \frac{f(\hat{\mathbf{h}}_S^{(i)},\mathbf{z}_i) - f(\hat{\mathbf{h}}_S,\mathbf{z}_i)}{m} + \frac{f(\hat{\mathbf{h}}_S,\mathbf{z}_i') - f(\hat{\mathbf{h}}_S^{(i)},\mathbf{z}_i')}{m}\\
&\quad + \left(F_S^{(i)}(\hat{\mathbf{h}}_S^{(i)}) - F_S^{(i)}(\hat{\mathbf{h}}_S)\right)\\
&\leq \frac{|f(\hat{\mathbf{h}}_S^{(i)},\mathbf{z}_i) - f(\hat{\mathbf{h}}_S,\mathbf{z}_i)|}{m} + \frac{|f(\hat{\mathbf{h}}_S,\mathbf{z}_i') - f(\hat{\mathbf{h}}_S^{(i)},\mathbf{z}_i')|}{m}\\
&\leq \frac{2L}{m}\left\|\hat{\mathbf{h}}_S^{(i)} - \hat{\mathbf{h}}_S\right\|
\end{aligned} \tag{7}$$

where the first inequality follows from the fact that $\hat{\mathbf{h}}_S^{(i)}$ is the minimizer of $F_S^{(i)}(\mathbf{h})$ and for the second inequality we use Lipschitz continuity. But from strong convexity of $F_S(\mathbf{h})$ and the fact that $\hat{\mathbf{h}}_S$ minimizes $F_S(\mathbf{h})$ we also have that

$$F_S(\hat{\mathbf{h}}_S^{(i)}) \geq F_S(\hat{\mathbf{h}}_S) + \tfrac{\lambda}{2}\left\|\hat{\mathbf{h}}_S^{(i)} - \hat{\mathbf{h}}_S\right\|^2. \tag{8}$$

Combining Equation (8) with Equation (7) we get $\left\|\hat{\mathbf{h}}_S^{(i)} - \hat{\mathbf{h}}_S\right\| \leq 4L/(\lambda m)$ and combining this with Lipschitz continuity of f we obtain that Equation (6) holds. Later on in this paper, we show that a stable ERM is sufficient for learnability. More formally, Equation (6) implies that the ERM is uniform-RO stability (Definition 4) with rate $\varepsilon_{\text{stable}}(m) = 4L^2/(\lambda m)$ and therefore Theorem 8 implies that the ERM is consistent with rate $\leq \varepsilon_{\text{stable}}(m)$, namely

$$\mathbb{E}_{S\sim\mathcal{D}^m}\left[F(\hat{\mathbf{h}}_S) - F^*\right] \leq \tfrac{4L^2}{\lambda m}\,.$$

Since the random variable in the expectation is non-negative, the theorem follows by Markov's inequality. ■

We now turn to the proof of Theorem 3.

Proof [Proof of Theorem 3] Let $r(\mathbf{h};\mathbf{z}) = \frac{\lambda}{2}\|\mathbf{h}\|^2 + f(\mathbf{h};\mathbf{z})$ and let $R(\mathbf{h}) = \mathbb{E}_{\mathbf{z}}[r(\mathbf{h},\mathbf{z})]$. Note that $\hat{\mathbf{h}}_\lambda$ is the empirical minimizer for the stochastic optimization problem defined by $r(\mathbf{h};\mathbf{z})$.

We apply Theorem 2 to $r(\mathbf{h};\mathbf{z})$, to this end note that since f is L-Lipschitz and $\forall \mathbf{h}\in\mathcal{H},\ \|\mathbf{h}\|\leq B$ we see that r is in fact $L+\lambda B$-Lipschitz. Applying Theorem 2, we see that

$$\tfrac{\lambda}{2}\left\|\hat{\mathbf{h}}_\lambda\right\|^2 + F(\hat{\mathbf{h}}_\lambda) = R(\hat{\mathbf{h}}_\lambda) \leq \inf_{\mathbf{h}} R(\mathbf{h}) + \frac{4(L+\lambda B)^2}{\delta\lambda m}$$

Now note that $\inf_{\mathbf{h}} R(\mathbf{h}) \leq \inf_{\mathbf{h}} F(\mathbf{h}) + \frac{\lambda}{2}B^2 = F^* + \frac{\lambda}{2}B^2$, and so we get that

$$\begin{aligned} F(\hat{\mathbf{h}}_\lambda) &\leq F^* + \frac{\lambda}{2}B^2 + \frac{4(L+\lambda B)^2}{\delta\lambda m} \\ &\leq F^* + \frac{\lambda}{2}B^2 + \frac{8L^2}{\delta\lambda m} + \frac{8\lambda B^2}{\delta m} \end{aligned}$$

Plugging in the value of λ given in the theorem statement we see that

$$F(\hat{\mathbf{h}}_\lambda) \leq F^* + 4\sqrt{\frac{L^2B^2}{\delta m}} + \frac{32}{\delta m}\sqrt{\frac{L^2B^2}{\delta m}}$$

This gives us the required bound. ■

From the above theorem, we see that regularization is essential for convex stochastic optimization. It is important to note that even for the strongly convex optimization problem in Theorem 2, where the ERM algorithm does work, it is not due to uniform convergence. To see this, consider augmenting the objective function $f^{(3)}(\cdot)$ from Equation (3) with a strongly convex term:

$$f^{(9)}(\mathbf{h};\mathbf{x},\alpha) = f^{(3)}(\mathbf{h};\mathbf{x},\alpha) + \frac{\lambda}{2}\|\mathbf{h}\|^2. \tag{9}$$

The modified objective $f^{(9)}(\cdot;\cdot)$ is λ-strongly convex and $(1+\lambda)$-Lipschitz over $\mathcal{H} = \{\mathbf{h} : \|\mathbf{h}\| \leq 1\}$ and thus satisfies the conditions of Theorem 2. Now, consider the same distribution over $z = (\mathbf{x},\alpha)$ used earlier: $\mathbf{x} = 0$ and α is an i.i.d. sequence of uniform zero/one Bernoulli variables. Recall that almost surely we have a coordinate j that is never "observed", namely such that $\forall_i \alpha_i[j] = 0$. Consider a vector $t\mathbf{e}_j$ of magnitude $0 < t \leq 1$ in the direction of this coordinate. We have that $F_S^{(9)}(t\mathbf{e}_j) = \frac{\lambda}{2}t^2$ (where $F_S^{(9)}(\cdot)$ is the empirical risk w.r.t. $f^{(9)}(\cdot)$) but $F^{(9)}(t\mathbf{e}_j) = \frac{1}{2}t + \frac{\lambda}{2}t^2$. Hence, letting $F^{(9)}(\cdot)$ denote the risk w.r.t. $f^{(9)}(\cdot)$, we have that $F^{(9)}(t\mathbf{e}_j) - F_S^{(9)}(t\mathbf{e}_j) = t/2$. In particular, we can set $t = 1$ and establish $\sup_{\mathbf{h}\in\mathcal{H}}(F^{(9)}(\mathbf{h}) - F_S^{(9)}(\mathbf{h})) \geq \frac{1}{2}$ regardless of the sample size.

We see then that the empirical averages $F_S^{(9)}(\mathbf{h})$ do *not* converge uniformly to their expectations. Moreover, the example above shows that there is no uniform convergence even in a local sense, namely over all hypotheses whose risk is close enough to F^*, or those close enough to the minimizer of $f^{(9)}(\mathbf{h};\mathbf{x},\alpha)$.

Finally, we note that the learning algorithm we have discussed here is mainly for pedagogical reasons. A different generic algorithm for stochastic convex optimization is already known in the literature, by combining Zinkevich's algorithm (Zinkevich, 2003) for online convex optimization, with an online-to-batch conversion (e.g., Cesa-Bianchi et al., 2004). While different than our algorithm, Shalev-Shwartz (2007) showed that Zinkevich's online learning algorithm can be viewed as approximate coordinate ascent optimization of the dual of the regularized problem Equation (5). Thus, this algorithm still uses the same mechanisms of regularization and stability. Also, we note that the algorithm also enjoys bounds which depend only logarithmically on $1/\delta$, while the bounds we have obtained above depend linearly on $1/\delta$. However, we suspect that the dependence on δ in Theorem 2 can be improved to $\log(1/\delta)$. For instance, such bounds has been obtained whenever the objective function is a generalized linear function of h (Sridharan et al., 2008).

4.3 How to Interpret Regularization: Uniform Convergence vs Stability

The technique of regularizing the objective function by adding a "bias" term is old and well known. In particular, adding $\|\mathbf{h}\|^2$ is the so-called Tikhonov Regularization technique, which has been known for more than half a century (see Tikhonov, 1943). However, the role of regularization in our case is very different than in familiar settings such as ℓ_2 regularization in SVMs and ℓ_1 regularization in LASSO. In those settings regularization serves to constrain our domain to a low-complexity domain (e.g., low-norm predictors), where

we rely on uniform convergence. In fact, almost all learning guarantees that we are aware of can be expressed in terms of some sort of uniform convergence.

In our case, constraining the norm of $\mathbf{h}$ does *not* ensure uniform convergence. Consider the example $f^{(3)}(\cdot)$ we have seen earlier. Even over a restricted domain $\mathcal{H}_r = \{\mathbf{h} : \|\mathbf{h}\| \leq r\}$, for arbitrarily small $r > 0$, the empirical averages $F_S(\mathbf{h})$ do *not* uniformly converge to $F(\mathbf{h})$. Furthermore, consider replacing the regularization term $\lambda \|\mathbf{h}\|^2$ with a constraint on the norm of $\|\mathbf{h}\|$, namely, solving the problem

$$\tilde{\mathbf{h}}_r = \arg\min_{\|\mathbf{h}\| \leq r} F_S(\mathbf{h})$$

We cannot solve the stochastic optimization problem by setting r in a distribution-independent way (i.e., without knowing the solution...). To see this, note that when $\mathbf{x} = 0$ a.s. we must have $r \to 0$ to ensure $F(\tilde{\mathbf{h}}_r) \to F^*$. However, if $\mathbf{x} = \mathbf{e}_1$ a.s., we must set $r \to 1$. No constraint will work for all distributions over $\mathbb{Z} = (\mathcal{X}, \alpha)$! This sharply contrasts with traditional uses of regularization, where learning guarantees are typically stated in terms of a constraint on the norm rather than in terms of a parameter such as λ, and adding a regularization term of the form $\frac{\lambda}{2} \|\mathbf{h}\|^2$ is viewed as a proxy for bounding the norm $\|\mathbf{h}\|$.

4.4 Contradiction to Vapnik?

In Section 3.1, we discussed how Vapnik showed that uniform convergence is in fact necessary for learnability with the ERM. At first glance, this might seem confusing in light of the examples presented above, where we have problems learnable with the ERM without uniform convergence whatsoever.

The solution for this apparent paradox is that our examples are not "strictly consistent" in Vapnik's sense. Recall that in order to exclude "trivial" cases, Vapnik defined strict consistency of empirical minimization as (in our notation):

$$\forall c \in \mathbb{R}, \quad \inf_{\mathbf{h}:F(\mathbf{h}) \geq c} F_S(\mathbf{h}) \longrightarrow \inf_{\mathbf{h}:F(\mathbf{h}) \geq c} F(\mathbf{h}) \ , \tag{10}$$

where the convergence is in probability. This condition indeed ensures that $F(\hat{\mathbf{h}}_S) \xrightarrow{P} F^*$. Vapnik's Key Theorem on Learning Theory (Vapnik, 1998, Theorem 3.1) then states that *strict* consistency of empirical minimization is equivalent to one-sided[2] uniform convergence. In the example presented above, even though Theorem 2 establishes $F^{(9)}(\hat{\mathbf{h}}_S) \xrightarrow{P} F^*$, the consistency isn't "strict" by the definition above. To see this, for any $c > 0$, consider the vector $t\mathbf{e}_j$ (where $\forall_i \alpha_i[j] = 0$) with $t = 2c$. We have $F^{(9)}(t\mathbf{e}_j) = \frac{1}{2}t + \frac{\lambda}{2}t^2 > c$ but $F_S^{(9)}(t\mathbf{e}_j) = \frac{\lambda}{2}t^2 = 2\lambda c^2$. Focusing on $\lambda = \frac{1}{2}$ we get:

$$\inf_{F^{(9)}(\mathbf{h}) \geq c} F_S^{(9)}(\mathbf{h}) \leq c^2$$

almost surely for any sample size m, violating the strict consistency requirement Equation (10).

We emphasize that stochastic convex optimization is far from "trivial" in that there is no dominating hypothesis that will always be selected. Although for convenience of analysis we took $\mathbf{x} = 0$, one should think of situations in which $\mathbf{x}$ is stochastic with an unknown distribution. This shows that uniform convergence is a sufficient, but not at all necessary, condition for consistency of empirical minimization in non-trivial settings.

5. Learnability in the General Learning Setting: the role of Stability

In the previous section, we have shown that in the General Learning Setting, it is possible for problems to be learnable without uniform convergence, in sharp contrast to previously considered settings. The key underlying mechanism which allowed us to learn is stability. In this section, we study the connection between learnability and stability in greater depth, and show that stability can in fact *characterize* learnability. Also, we will see how various "common knowledge facts", which we usually take for granted and are based on a

2. "One-sided" meaning requiring only $\sup(F(\mathbf{h}) - F_S(\mathbf{h})) \longrightarrow 0$, rather then $\sup|F(\mathbf{h}) - F_S(\mathbf{h})| \longrightarrow 0$.

"uniform convergence equivalent to learnability" assumption, do not hold in the General Learning Setting, and things can be much more delicate.

We will refer to settings where learnability is equivalent to uniform convergence as "supervised classification" settings. While supervised classification does not encompass all settings where this equivalence holds, most equivalence results refer to it either explicitly or implicitly (by reduction to a classification problem).

5.1 Stability : Definitions

We start by giving the exact definition of the stability notions that we will use. As discussed earlier, there are many possible stability measures, some of which can be used to obtain results of a similar flavor to the ones below. The definition we use seems to be the most convenient for the goal of characterizing learnability in the General Learning Setting. In Appendix A, we provide a few illustrating examples to the subtle differences that can arise from slight variations in the stability measure.

Our two stability notions are based on replacing one of the training sample instances. For a sample S of size m, let $S^{(i)} = \{\mathbf{z}_1, ..., \mathbf{z}_{i-1}, \mathbf{z}'_i, \mathbf{z}_{i+1}, ..., \mathbf{z}_m\}$ be a sample obtained by replacing the i-th observation of S with some different instance $\mathbf{z}'_i$. When not discussed explicitly, the nature of how $\mathbf{z}'_i$ is obtained should be obvious from context.

Definition 4 *A rule* $\mathbf{A}$ *is* **uniform-RO stable**[3] *with rate* $\varepsilon_{\text{stable}}(m)$*, if for all possible* $S^{(i)}$ *and any* $\mathbf{z}' \in \mathcal{Z}$*,*

$$\frac{1}{m}\sum_{i=1}^{m}\left|f(\mathbf{A}(S^{(i)});\mathbf{z}') - f(\mathbf{A}(S);\mathbf{z}')\right| \leq \varepsilon_{\text{stable}}(m).$$

Definition 5 *A rule* $\mathbf{A}$ *is* **average-RO stable** *with rate* $\varepsilon_{\text{stable}}(m)$ *under distributions* $\mathcal{D}$ *if*

$$\left|\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{S\sim\mathcal{D}^m,(\mathbf{z}'_1,...,\mathbf{z}'_m)\sim\mathcal{D}^m}\left[f(\mathbf{A}(S^{(i)});\mathbf{z}'_i) - f(\mathbf{A}(S);\mathbf{z}'_i)\right]\right| \leq \varepsilon_{\text{stable}}(m).$$

Note that this definition corresponds to assuming that the expected empirical risk of the learning rule converges to the expected risk - see Lemma 11.

We say that a rule is *universally* stable with rate $\varepsilon_{\text{stable}}(m)$, if the stability property holds with rate $\varepsilon_{\text{stable}}(m)$ for all distributions.

Claim 6 *Uniform-RO stability with rate* $\varepsilon_{\text{stable}}(m)$ *implies average-RO stability with rate* $\varepsilon_{\text{stable}}(m)$*.*

5.2 Characterizing Learnability : Main Results

Our overall goal is to characterize learnable problems (namely, problems for which there exists a universally consistent learning rule, as in Equation (2)). That means finding some condition which is both *necessary* and *sufficient* for learnability. In the uniform convergence setting, such a condition is the stability of the ERM (under any of several possible stability measures, including both variants of RO-stability defined above). This is still sufficient for learnability in the General Learning Setting, but far from being necessary, as we have seen in Section 4.

The most important result in this section is a condition which is necessary and sufficient for learnability in the General Learning Setting:

Theorem 7 *A learning problem is learnable if and only if there exists a uniform-RO stable, universally AERM learning rule.*

In particular, if there exists a $\varepsilon_{\text{cons}}(m)$*-universally consistent rule, then there exists a rule that is* $\varepsilon_{\text{stable}}(m)$*-uniform-RO stable and universally* $\varepsilon_{\text{erm}}(m)$*-AERM where:*

$$\varepsilon_{\text{erm}}(m) = 3\varepsilon_{\text{cons}}(m^{1/4}) + \tfrac{8B}{\sqrt{m}} \ ,$$
$$\varepsilon_{\text{stable}}(m) = \tfrac{2B}{\sqrt{m}}.$$

3. RO is short for "replace-one".

In the opposite direction, if a learning rule is $\varepsilon_{\text{stable}}(m)$*-uniform-RO stable and universally* $\varepsilon_{\text{erm}}(m)$*-AERM, then it is universally consistent with rate*

$$\varepsilon_{\text{cons}}(m) \leq \varepsilon_{\text{stable}}(m) + \varepsilon_{\text{erm}}(m)$$

Thus, while we have seen in Section 4 that the ERM rule might fail for learning problems which are in fact learnable, there is always an AERM rule which will work. In other words, when designing learning rules, we might need to look beyond empirical risk minimization, but not beyond AERM learning rules. On the downside, we must choose our AERM carefully, since not any AERM will work. This contrasts with supervised classification, where any AERM will work if the problem is learnable at all.

How do we go about proving this assertion? The easier part is showing sufficiency. Namely, that a stable AERM must be consistent (and generalizing). In fact, this holds both separately for any particular distribution $\mathcal{D}$s, and uniformly over all distributions:

Theorem 8 *If a rule is an AERM with rate* $\varepsilon_{\text{erm}}(m)$ *and average-RO stable (or uniform-RO stable) with rate* $\varepsilon_{\text{stable}}(m)$ *under* $\mathcal{D}$*, then it is consistent and generalizes under* $\mathcal{D}$ *with rates*

$$\varepsilon_{\text{cons}}(m) \leq \varepsilon_{\text{stable}}(m) + \varepsilon_{\text{erm}}(m)$$
$$\varepsilon_{\text{gen}}(m) \leq \varepsilon_{\text{stable}}(m) + 2\varepsilon_{\text{erm}}(m) + \tfrac{2B}{\sqrt{m}}$$

The second part of Theorem 7 follows as a direct corollary. We note that close variants of Theorem 8 has already appeared in previous literature (e.g., Mukherjee et al., 2006 and Rakhlin et al., 2005).

The harder part is showing that a uniform-RO stable AERM is *necessary* for learnability. This is done in several steps.

First, we show that consistent AERMs have to be average-RO stable:

Theorem 9 *For an AERM, the following are equivalent:*
- *Universal average-RO stability.*
- *Universal consistency.*
- *Universal generalization.*

The exact conversion rate of Theorem 9 is specified in the corresponding proof (Section 5.3), and are all polynomial. In particular, an $\varepsilon_{\text{cons}}$-universal consistent ε_{erm}-AERM is average-RO stable with rate

$$\varepsilon_{\text{stable}}(m) \leq \varepsilon_{\text{erm}}(m) + 3\varepsilon_{\text{cons}}(m^{1/4}) + \tfrac{4B}{\sqrt{m}}.$$

Next, we show that if we seek universally consistent and generalizing learning rules, then we must consider only AERMs:

Theorem 10 *If a rule* $\mathbf{A}$ *is universally consistent with rate* $\varepsilon_{\text{cons}}(m)$ *and generalizing with rate* $\varepsilon_{\text{gen}}(m)$*, then it is universally an AERM with rate*

$$\varepsilon_{\text{erm}}(m) \leq \varepsilon_{\text{gen}}(m) + 3\varepsilon_{\text{cons}}(m^{1/4}) + \frac{4B}{\sqrt{m}}$$

Now, recall that learnability is defined as the existence of some universally consistent learning rule. Such a rule might not be generalizing, stable or even an AERM (see example 2 below). However, it turns out that if a universally consistent learning rule exist, then there is *another* learning rule for the same problem, which is generalizing (Lemma 20). Thus, by Theorems 9-10, this rule must also be average-RO stable AERM. In fact, by another application of Lemma 20, such an AERM must also be uniform-RO stable, leading to Theorem 7.

5.3 Detailed Results and Proofs

We first establish that for AERMs, average-RO stability and generalization are equivalent.

5.3.1 EQUIVALENCE OF STABILITY AND GENERALIZATION

It will be convenient to work with a weaker version of generalization as an intermediate step: We say a rule **A on-average generalizes** with rate $\varepsilon_{\text{oag}}(m)$ under distribution $\mathcal{D}$ if for all m,

$$|\mathbb{E}_{S\sim\mathcal{D}^m}[F(\mathbf{A}(S)) - F_S(\mathbf{A}(S))]| \leq \varepsilon_{\text{oag}}(m). \tag{11}$$

It is straightforward to see that generalization implies on-average generalization with the same rate. We show that for AERMs, the converse is also true, and also that on-average generalization is equivalent to average-RO stability. This establishes the equivalence between generalization and average-RO stability (for AERMs).

Lemma 11 (on-average generalization $\Leftrightarrow$ average-RO stability) *If* $\mathbf{A}$ *is on-average generalizing with rate* $\varepsilon_{\text{oag}}(m)$ *then it is average-RO stable with rate* $\varepsilon_{\text{oag}}(m)$. *If* $\mathbf{A}$ *is average-RO stable with rate* $\varepsilon_{\text{stable}}(m)$ *then it is on-average generalizing with rate* $\varepsilon_{\text{stable}}(m)$.

Proof For any i, $\mathbf{z}_i$ and $\mathbf{z}'_i$ are both drawn i.i.d. from $\mathcal{D}$, we have that

$$\mathbb{E}_{S\sim\mathcal{D}^m}[f(\mathbf{A}(S);\mathbf{z}_i)] = \mathbb{E}_{S\sim\mathcal{D}^m,\mathbf{z}'_i\sim\mathcal{D}}\left[f(\mathbf{A}(S^{(i)});\mathbf{z}'_i)\right].$$

Hence,

$$\begin{aligned}
\mathbb{E}_{S\sim\mathcal{D}^m}[F_S(\mathbf{A}(S))] &= \mathbb{E}_{S\sim\mathcal{D}^m}\left[\frac{1}{m}\sum_{i=1}^m f(\mathbf{A}(S);\mathbf{z}_i)\right] \\
&= \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{S\sim\mathcal{D}^m}[f(\mathbf{A}(S);\mathbf{z}_i)] \\
&= \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{S\sim\mathcal{D}^m,\mathbf{z}'_i\sim\mathcal{D}}\left[f(\mathbf{A}(S^{(i)});\mathbf{z}'_i)\right]
\end{aligned}$$

Also note that $F(\mathbf{A}(S)) = \mathbb{E}_{\mathbf{z}'_i\sim\mathcal{D}}[f(\mathbf{A}(S);\mathbf{z}'_i)] = \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\mathbf{z}'_i\sim\mathcal{D}}[f(\mathbf{A}(S);\mathbf{z}'_i)]$. Hence we can conclude that

$$\mathbb{E}_{S\sim\mathcal{D}^m}[F(\mathbf{A}(S)) - F_S(\mathbf{A}(S))] = \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{S\sim\mathcal{D}^m,(\mathbf{z}'_1,\ldots,\mathbf{z}'_m)\sim\mathcal{D}^m}\left[f(\mathbf{A}(S);\mathbf{z}'_i) - f(\mathbf{A}(S^{(i)});\mathbf{z}'_i)\right]$$

Hence we have the required result. ■

For the next result, we will need the following two short utility lemmas.

Utility Lemma 12 *For i.i.d.* X_i, $|X_i| \leq B$ *and* $X = \frac{1}{m}\sum_{i=1}^m X_i$ *we have* $\mathbb{E}[|X - \mathbb{E}[X]|] \leq B/\sqrt{m}$.

Proof $\mathbb{E}[|X - \mathbb{E}[X]|] \leq \sqrt{\mathbb{E}\left[|X - \mathbb{E}[X]|^2\right]} \leq \sqrt{\text{Var}[X]} = \sqrt{\text{Var}[X_i]/m} \leq B/\sqrt{m}$. ■

Utility Lemma 13 *Let* X, Y *be random variables s.t.* $X \leq Y$ *almost surely. Then* $\mathbb{E}[|X|] \leq |\mathbb{E}[X]| + 2\mathbb{E}[|Y|]$.

Proof

$$\mathbb{E}[|X|] = \mathbb{E}[|(Y-X) - Y|] \leq \mathbb{E}[Y - X] + \mathbb{E}[|Y|] \leq |\mathbb{E}[X]| + 2|\mathbb{E}[Y]|.$$

■

Lemma 14 (AERM + on-average generalization ⇒ generalization) *If* $\mathbf{A}$ *is an AERM with rate* $\varepsilon_{\text{erm}}(m)$ *and on-average generalizes with rate* $\varepsilon_{\text{oag}}(m)$ *under* $\mathcal{D}$*, then* $\mathbf{A}$ *generalizes with rate* $\varepsilon_{\text{oag}}(m)+2\varepsilon_{\text{erm}}(m)+\frac{2B}{\sqrt{m}}$ *under* $\mathcal{D}$*.*

Proof Recall that $F^* = \inf_{\mathbf{h}\in\mathcal{H}} F(\mathbf{h})$. For an arbitrarily small $\nu > 0$, let $\mathbf{h}_\nu$ be a fixed hypothesis such that $F(\mathbf{h}_\nu) \leq F^* + \nu$. Using respective optimalities of $\hat{\mathbf{h}}_S$ and F^* we can bound:

$$\begin{aligned}
&F_S(\mathbf{A}(S)) - F(\mathbf{A}(S)) \\
&= F_S(\mathbf{A}(S)) - F_S(\hat{\mathbf{h}}_S) + F_S(\hat{\mathbf{h}}_S) - F_S(\mathbf{h}_\nu) + F_S(\mathbf{h}_\nu) - F(\mathbf{h}_\nu) + F(\mathbf{h}_\nu) - F(\mathbf{A}(S)) \\
&\leq F_S(\mathbf{A}(S)) - F_S(\hat{\mathbf{h}}_S) + F_S(\mathbf{h}_\nu) - F(\mathbf{h}_\nu) + \nu = Y_\nu
\end{aligned}$$

Where the final equality defines a new random variable Y_ν. By Lemma 12 and the AERM guarantee we have $\mathbb{E}[|Y_\nu|] \leq \varepsilon_{\text{erm}}(m) + B/\sqrt{m} + \nu$. From Lemma 13 we can conclude that

$$\mathbb{E}\left[\left|F_S(\mathbf{A}(S)) - F(\mathbf{A}(S))\right|\right] \leq \left|\mathbb{E}\left[F_S(\mathbf{A}(S)) - F(\mathbf{A}(S))\right]\right| + 2\mathbb{E}\left[|Y_\nu|\right] \leq \varepsilon_{\text{oag}}(m) + 2\varepsilon_{\text{erm}}(m) + \tfrac{2B}{\sqrt{m}} + \nu.$$

Notice that the l.h.s. is a fixed quantity which does not depend on ν. Therefore, we can take ν in the r.h.s. to zero, and the result follows. ■

Combining Lemma 11 and Lemma 14, we have now **established the stability↔generalization parts of Theorem 8 and Theorem 9** (in fact, even a slightly stronger converse than in Theorem 9, as it does not require universality).

5.3.2 A SUFFICIENT CONDITION FOR CONSISTENCY

It is fairly straightforward to see that generalization (or even on-average generalization) of an AERM implies its consistency:

Lemma 15 (AERM+generalization⇒consistency) *If* $\mathbf{A}$ *is AERM with rate* $\varepsilon_{\text{erm}}(m)$ *and it on-average generalizes with rate* $\varepsilon_{\text{oag}}(m)$ *under* $\mathcal{D}$ *then it is consistent with rate* $\varepsilon_{\text{oag}}(m) + \varepsilon_{\text{erm}}(m)$ *under* $\mathcal{D}$*.*

Proof For any $\nu > 0$, let $\mathbf{h}_\nu$ be a hypothesis such that $F(\mathbf{h}_\nu) \leq F^* + \nu$. We have

$$\begin{aligned}
&\mathbb{E}\left[F(\mathbf{A}(S)) - F^*\right] = \mathbb{E}\left[F(\mathbf{A}(S)) - F_S(\mathbf{h}_\nu) + \nu\right] \\
&= \mathbb{E}\left[F(\mathbf{A}(S)) - F_S(\mathbf{A}(S))\right] + \mathbb{E}\left[F_S(\mathbf{A}(S)) - F_S(\mathbf{h}_\nu)\right] + \nu \\
&\leq \mathbb{E}\left[F(\mathbf{A}(S)) - F_S(\mathbf{A}(S))\right] + \mathbb{E}\left[F_S(\mathbf{A}(S)) - F_S(\hat{\mathbf{h}}_S)\right] + \nu \\
&\leq \varepsilon_{\text{oag}}(m) + \varepsilon_{\text{erm}}(m) + \nu.
\end{aligned}$$

Since this upper bound holds for any ν, we can take ν to zero, and the result follows. ■

Combined with the results of Lemma 11, this completes the **proof of Theorem 8** and the **stability → consistency and generalization → consistency parts of Theorem 9**.

5.3.3 CONVERSE DIRECTION

Lemma 11 already provides a converse result, establishing that stability is necessary for generalization. However, as it will turn out, in order to establish that stability is also necessary for *universal consistency*, we must prove that universal consistency of an AERM implies *universal* generalization. The assumption of *universal* consistency for the AERM is crucial here: mere consistency of an AERM with respect to a specific distribution does *not* imply generalization nor stability with respect to that distribution. The following example briefly illustrates this point.

Example 1 *There exists a learning problem and a distribution on the instance space, such that the ERM (or any AERM) is consistent with rate* $\varepsilon_{\text{cons}}(m) = 0$*, but does not generalize and is not average-RO stable (namely,* $\varepsilon_{\text{gen}}(m), \varepsilon_{\text{stable}}(m) = \Omega(1)$*).*

Proof Let the instance space be $[0,1]$, the hypothesis space consist of all finite subsets of $[0,1]$, and define the objective function as $f(\mathbf{h}, z) = \mathbb{1}_{[z \notin \mathbf{h}]}$. Consider any continuous distribution on the instance space. Since the underlying distribution $\mathcal{D}$ is continuous, we have $F(\mathbf{h}) = 1$ for any hypothesis h. Therefore, any learning rule (including any AERM) will be consistent with $F(\mathbf{A}(S)) = 1$. On the other hand, the ERM here always achieves $F_S(\hat{\mathbf{h}}_S) = 0$, so any AERM cannot generalize, or even on-average-generalize (by Lemma 14), hence cannot be average-RO stable (by Lemma 11). ■

The main tool we use to prove our desired converse result is the following lemma. It is here that we crucially use the universal consistency assumption (i.e., consistency with respect to *any* distribution). Intuitively, it states that if a problem is learnable at all, then although the ERM rule might fail, its empirical risk is a consistent estimator of the minimal achievable risk.

Lemma 16 (Main Converse Lemma) *If a problem is learnable, namely there exists a universally consistent rule* $\mathbf{A}$ *with rate* $\varepsilon_{\text{cons}}(m)$*, then under any distribution,*

$$\mathbb{E}\left[\left|F_S(\hat{\mathbf{h}}_S) - F^*\right|\right] \leq \varepsilon_{\text{emp}}(m) \qquad \textit{where} \tag{12}$$

$$\varepsilon_{\text{emp}}(m) = 2\varepsilon_{\text{cons}}(m') + \tfrac{2B}{\sqrt{m}} + \tfrac{2Bm'^2}{m}$$

for any m' *such that* $2 \leq m' \leq m/2$.

Proof Let $I = \{I_1, \ldots, I_{m'}\}$ be a random sample of m' indexes in the range $1..m$ where each I_i is independently uniformly distributed, and I is independent of S. Let $S' = \{z_{I_i}\}_{i=1}^{m'}$, that is, a sample of size m' drawn from the uniform distribution over samples in S (with replacements). We first bound the probability that I has no repeated indexes ("duplicates"):

$$\mathbb{P}\left[I \text{ has duplicates}\right] \leq \frac{\sum_{i=1}^{m'}(i-1)}{m} \leq \frac{m'^2}{2m} \tag{13}$$

Conditioned on not having duplicates in I, the sample S' is actually distributed according to $\mathcal{D}^{m'}$, that is, can be viewed as a sample from the original distribution. We therefore have by universal consistency:

$$\mathbb{E}\left[\left|F(\mathbf{A}(S')) - F^*\right| \,\middle|\, \text{no dups}\right] \leq \varepsilon_{\text{cons}}(m') \tag{14}$$

But viewed as a sample drawn from the uniform distribution over instances in S, we also have:

$$\mathbb{E}_{S'}\left[\left|F_S(\mathbf{A}(S')) - F_S(\hat{\mathbf{h}}_S)\right|\right] \leq \varepsilon_{\text{cons}}(m') \tag{15}$$

Conditioned on having no duplications in I, the set of those samples in S not chosen by I (i.e., $S \setminus S'$) is independent of S', and $|S \setminus S'| = m - m'$, and so by Lemma 12:

$$\mathbb{E}_S\left[\left|F(\mathbf{A}(S')) - F_{S\setminus S'}(\mathbf{A}(S'))\right|\right] \leq \frac{B}{\sqrt{m-m'}} \tag{16}$$

Finally, if there are no duplicates, then for any hypothesis, and in particular for $\mathbf{A}(S')$ we have:

$$\left|F_S(\mathbf{A}(S')) - F_{S\setminus S'}(\mathbf{A}(S'))\right| \leq \frac{2Bm'}{m} \tag{17}$$

Combining Equation (14),Equation (15),Equation (16) and Equation (17), accounting for a maximal discrepancy of B when we do have duplicates, and assuming $2 \leq m' \leq m/2$, we get the desired bound. ■

Equipped with Lemma 16, we are now ready to show that universal consistency of an AERM implies universal generalization and that any universally consistent and generalizing rule must be an AERM. What we show is actually a bit stronger: that if a problem is learnable, and so Lemma 16 holds, then for any distribution $\mathcal{D}$ separately, consistency of an AERM under $\mathcal{D}$ implies generalization under $\mathcal{D}$ and also any consistent and generalizing rule under $\mathcal{D}$ must be an AERM.

Lemma 17 (learnable+AERM+consistent⇒generalizing) *If Equation (12) in Lemma 16 holds with rate $\varepsilon_{\text{emp}}(m)$, and $\mathbf{A}$ is an ε_{erm}-AERM and $\varepsilon_{\text{cons}}$-consistent under $\mathcal{D}$, then it is generalizing under $\mathcal{D}$ with rate $\varepsilon_{\text{emp}}(m)+\varepsilon_{\text{erm}}(m)+\varepsilon_{\text{cons}}(m)$.*

Proof

$$\begin{aligned}\mathbb{E}\left[\left|F_S(\mathbf{A}(S))-F(\mathbf{A}(S))\right|\right] &\leq \mathbb{E}\left[\left|F_S(\mathbf{A}(S))-F_S(\hat{\mathbf{h}}_S)\right|\right]+\mathbb{E}\left[\left|F^*-F(\mathbf{A}(S))\right|\right]+\mathbb{E}\left[\left|F_S(\hat{\mathbf{h}}_S)-F^*\right|\right] \\ &\leq \varepsilon_{\text{erm}}(m)+\varepsilon_{\text{cons}}(m)+\varepsilon_{\text{emp}}(m)\ .\end{aligned}$$

■

Lemma 18 (learnable+consistent+generalizing⇒AERM) *If Equation (12) in Lemma 16 holds with rate $\varepsilon_{\text{emp}}(m)$, and $\mathbf{A}$ is $\varepsilon_{\text{cons}}$-consistent and ε_{gen}-generalizing under $\mathcal{D}$, then it is AERM under $\mathcal{D}$ with rate $\varepsilon_{\text{emp}}(m)+\varepsilon_{\text{gen}}(m)+\varepsilon_{\text{cons}}(m)$.*

Proof

$$\begin{aligned}\mathbb{E}\left[\left|F_S(\mathbf{A}(S))-F_S(\hat{\mathbf{h}}_S)\right|\right] &\leq \mathbb{E}\left[\left|F_S(\mathbf{A}(S))-F(\mathbf{A}(S))\right|\right]+\mathbb{E}\left[\left|F(\mathbf{A}(S))-F^*\right|\right]+\mathbb{E}\left[\left|F^*-F_S(\hat{\mathbf{h}}_S)\right|\right] \\ &\leq \varepsilon_{\text{gen}}(m)+\varepsilon_{\text{cons}}(m)+\varepsilon_{\text{emp}}(m)\ .\end{aligned}$$

■

Lemma 17 establishes that universal consistency of an AERM implies universal generalization, and thus **completes the proof of Theorem 9**. Lemma 18 **establishes Theorem 10**. To get the rates in Section 5.2, we use $m' = m^{1/4}$ in Lemma 16.

Lemma 15, Lemma 17 and Lemma 18 together establish an interesting relationship:

Corollary 19 *For a (universally) learnable problem, for any distribution $\mathcal{D}$ and learning rule $\mathbf{A}$, any two of the following imply the third :*

- $\mathbf{A}$ *is an AERM under $\mathcal{D}$.*
- $\mathbf{A}$ *is consistent under $\mathcal{D}$.*
- $\mathbf{A}$ *generalizes under $\mathcal{D}$.*

Note, however, that any one property by itself is possible, even universally:

- In Section 4.1, we have discussed an example where the ERM learning rule is neither consistent nor generalizing, despite the problem being learnable.
- In the next subsection (Example 2) we demonstrate a universally consistent learning rule which is neither generalizing nor an AERM.
- A rule returning a fixed hypothesis always generalizes, but of course need not be consistent nor an AERM.

In contrast, for learnable supervised classification problems, it is not possible for a learning rule to be just universally consistent, without being an AERM and without generalization. Nor is it possible for a learning rule to be a universal AERM for a learnable problem, without being generalizing and consistent.

Corollary 19 can also provide a *certificate* of non-learnability. In other words, for the problem in Example 1 we show a specific distribution for which there is a consistent AERM that does not generalize. We can conclude that there is *no* universally consistent learning rule for the problem, otherwise the corollary is violated.

5.3.4 EXISTENCE OF A STABLE RULE

Theorem 9 and Theorem 10, which we just completed proving, already establish that for AERMs, universal consistency is equivalent to universal average-RO stability. Existence of a universally average-RO stable AERM is thus sufficient for learnability. In order to prove that it is also necessary, it is enough to show that existence of a universally consistent learning rule implies existence of a universally consistent AERM. This AERM must then be average-RO stable by Theorem 9.

We actually show how to transform a consistent rule to a consistent and generalizing rule (Lemma 20 below). If this rule is universally consistent, then by Lemma 18 we can then conclude it must be an AERM, and by Lemma 11 it must be average-RO stable.

Lemma 20 *For any rule* $\mathbf{A}$ *there exists a rule* $\mathbf{A}'$*, such that:*

- $\mathbf{A}'$ *universally generalizes with rate* $\frac{3B}{\sqrt{m}}$.
- *For any* $\mathcal{D}$*, if* $\mathbf{A}$ *is* $\varepsilon_{\text{cons}}$*-consistent under* $\mathcal{D}$ *then* $\mathbf{A}'$ *is* $\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)$ *consistent under* $\mathcal{D}$.
- $\mathbf{A}'$ *is uniformly-RO-stable with rate* $\frac{2B}{\sqrt{m}}$.

Proof For a sample S of size m, let S' be a sub-sample consisting of some $\lfloor\sqrt{m}\rfloor$ observation in S. To simplify the presentation, assume that $\lfloor\sqrt{m}\rfloor$ is an integer. Define $\mathbf{A}'(S) = \mathbf{A}(S')$. That is, $\mathbf{A}'$ applies A to only $\sqrt{m}$ of the observation in S.

$\mathbf{A}'$ *generalizes:* We can decompose:

$$F_S(\mathbf{A}(S')) - F(\mathbf{A}(S')) = \tfrac{1}{\sqrt{m}}(F_{S'}(\mathbf{A}(S')) - F(\mathbf{A}(S'))) + (1-\tfrac{1}{\sqrt{m}})(F_{S\setminus S'}(\mathbf{A}(S')) - F(\mathbf{A}(S')))$$

The first term can be bounded by $2B/\sqrt{m}$. As for the second term, $S \setminus S'$ is statistically independent of S' and so we can use Lemma 12 to bound its expected magnitude to obtain:

$$\mathbb{E}\left[\left|F_S(\mathbf{A}(S')) - F(\mathbf{A}(S'))\right|\right] \leq \tfrac{2B}{\sqrt{m}} + (1-\tfrac{1}{\sqrt{m}})\tfrac{B}{\sqrt{m-\sqrt{m}}} \leq \tfrac{3B}{\sqrt{m}}$$

$\mathbf{A}'$ *is consistent:* If $\mathbf{A}$ is consistent, then:

$$\mathbb{E}\left[F(\mathbf{A}'(S)) - \inf_{\mathbf{h}\in\mathcal{H}} F(\mathbf{h})\right] = \mathbb{E}\left[F(\mathbf{A}(S')) - \inf_{\mathbf{h}\in\mathcal{H}} F(\mathbf{h})\right] \leq \varepsilon_{\text{cons}}(\sqrt{m})$$

$\mathbf{A}'$ *is uniformly-RO-stable:* Since $\mathbf{A}'$ only uses the first $\sqrt{m}$ samples of S, for any $i > \sqrt{m}$ we have $\mathbf{A}'(S^{(i)}) = \mathbf{A}'(S)$ and so:

$$\frac{1}{m}\sum_{i=1}^{m}\left|f(\mathbf{A}'(S^{(i)});\mathbf{z}') - f(\mathbf{A}'(S);\mathbf{z}')\right| = \frac{1}{m}\sum_{i=1}^{\sqrt{m}}\left|f(\mathbf{A}'(S^{(i)});\mathbf{z}') - f(\mathbf{A}'(S);\mathbf{z}')\right| \leq \frac{2B}{\sqrt{m}}$$

■

Proof of Converse in Theorem 7 If there exists a universally consistent rule with rate $\varepsilon_{\text{cons}}(m)$, by Lemma 20 there exists $\mathbf{A}'$ which is $\varepsilon_{\text{cons}}(\sqrt{m})$- universally consistent, $\frac{2B}{\sqrt{m}}$-generalizing and $\frac{2B}{\sqrt{m}}$-uniformly-RO-stable.

Further by Lemma 18 and Lemma 16 (with $m' = m^{1/4}$), we can conclude that A' is ε_{erm}-universally AERM where,

$$\varepsilon_{\text{erm}}(m) \leq 3\varepsilon_{\text{cons}}(m^{1/4}) + \frac{8B}{\sqrt{m}}.$$

Hence we get the specified rate for the converse direction. To see that if there exists a rule that is a universal AERM and stable it is consistent, we simply use Lemma 15.

As a final note, the following example shows that while learnability is equivalent to the existence of stable and consistent AERM's (Theorem 7 and Theorem 9), there might still exist other learning rules, which are neither stable, nor generalize, nor AERM's. In this sense, our results characterize learnability, but do not characterize all learning rules which "work".

Example 2 *There exists a learning problem with a universally consistent learning rule, which is not average-RO stable, generalizing nor an AERM.*

Proof Let the instance space be $[0,1]$. Let the hypothesis space consist of all finite subsets of $[0,1]$, and the objective function be the indicator function $f(\mathbf{h},z) = \mathbb{1}_{[z\in\mathbf{h}]}$. Consider the following learning rule: given a sample $S \subseteq [0,1]$, the learning rule checks if there are any two identical instances in the sample. If so, the learning rule returns the empty set $\emptyset$. Otherwise, it returns the sample.

Consider any continuous distribution on $[0,1]$. In that case, the probability of having two identical instances is 0. Therefore, the learning rule always returns a countable non-empty set $\mathbf{A}(S)$, with $F_S(\mathbf{A}(S)) = 1$, while $F_S(\emptyset) = 0$ (so it is not an AERM) and $F(\mathbf{A}(S)) = 0$ (so it does not generalize). Also, $f(\mathbf{A}(S), z_i) = 1$ while $f(\mathbf{A}(S^{(i)}), z_i) = 0$ with probability 1, so it is not average-RO stable either.

However, the learning rule is universally consistent. If the underlying distribution is continuous on $[0,1]$, then the returned hypothesis is S, which is countable hence , $F(S) = 0 = \inf_{\mathbf{h}} F(\mathbf{h})$. For discrete distributions, let M_1 denote the proportion of instances in the sample which appear exactly once, and let M_0 be the probability mass of instances which did not appear in the sample. Using (McAllester and Schapire, 2000, Theorem 3), we have that for any δ, it holds with probability at least $1-\delta$ over a sample of size m that

$$|M_0 - M_1| \leq O\left(\frac{\log(m/\delta)}{\sqrt{m}}\right),$$

uniformly for any discrete distribution. If this occurs, then either $M_1 < 1$, or $M_0 \geq 1 - O(\log(m/\delta)/\sqrt{m})$. But in the first event, we get duplicate instances in the sample, so the returned hypothesis is the optimal $\emptyset$, and in the second case, the returned hypothesis is the sample, which has a total probability mass of at most $O(\log(m/\delta)/\sqrt{m})$, and therefore $F(\mathbf{A}(S)) \leq O(\log(m/\delta)/\sqrt{m})$. As a result, regardless of the underlying distribution, with probability of at least $1-\delta$ over the sample,

$$F(\mathbf{A}(S)) \leq O\left(\frac{\log(m/\delta)}{\sqrt{m}}\right).$$

Since the r.h.s. converges to 0 with m for any δ, it is easy to see that the learning rule is universally consistent. ■

6. Randomization, Convexification, and a Generic Learning Algorithm

The strongest result we were able to obtain for characterizing learnability so far is Theorem 7, which stated that a problem is learnable if and only if there exists a universally uniform-RO stable AERM. In fact, this result was obtained under the assumption that the learning rule $\mathbf{A}$ is deterministic: given a fixed sample S, $\mathbf{A}$ returns a single specific hypothesis $\mathbf{h}$. However, we might relax this assumption and also consider *randomized* learning rules: given any fixed S, $\mathbf{A}(S)$ returns a distribution over the hypothesis class $\mathcal{H}$.

With this relaxation, we will see that we can obtain a stronger version of Theorem 7, and even provide a generic learning algorithm (at least for computationally unbounded learners) which successfully learns any learnable problem.

6.1 Stronger Results with Randomized Learning Rules

For simplicity, we will override the notations $f(\mathbf{A}(S),\mathbf{z})$, $F(\mathbf{A}(S))$ and $F_S(\mathbf{A}(S))$ to mean $\mathbb{E}_{\mathbf{h}\sim\mathbf{A}(S)}[f(\mathbf{h},\mathbf{z})]$, $\mathbb{E}_{\mathbf{h}\sim\mathbf{A}(S)}[F(\mathbf{h})]$ and $\mathbb{E}_{\mathbf{h}\sim\mathbf{A}(S)}[F_S(\mathbf{h})]$. In other words, $\mathbf{A}$ returns a distribution over $\mathcal{H}$ and $f(\mathbf{A}(S),\mathbf{z})$ for some fixed $S,\mathbf{z}$ is the expected loss of a random hypothesis picked according to that distribution, with respect to $\mathbf{z}$. Similarly, $F(\mathbf{A}(S))$ for some fixed S is the expected generalization error, and $F_S(\mathbf{A}(S))$ is the expected empirical risk on the fixed sample S. With this slight abuse of notation, all our previous definitions hold. For instance, we still define a learning rule $\mathbf{A}$ to be consistent with rate $\varepsilon_{\text{cons}}(m)$ if $\mathbb{E}_{S\sim\mathcal{D}^m}[F(\mathbf{A}(S))-F^*]\leq\varepsilon_{\text{cons}}(m)$, only now we actually mean

$$\mathbb{E}_{S\sim\mathcal{D}^m}\left[\mathbb{E}_{\mathbf{h}\sim\mathbf{A}(S)}[F(\mathbf{h})-F^*]\right]\leq\varepsilon_{\text{cons}}(m).$$

The definitions for AERM, generalization etc. also hold with this subtle change in meaning.

An alternative way to view randomization is as a method to *linearize* the learning problem. In other words, randomization implicitly replaces the arbitrary hypothesis class $\mathcal{H}$ by the space of probability distributions over $\mathcal{H}$,

$$\mathcal{M}=\left\{\alpha:\mathcal{H}\to[0,1]\ \text{ s.t. }\int\alpha[\mathbf{h}]=1\right\},$$

and replaces the arbitrary function $f(\mathbf{h};\mathbf{z})$ by a *linear* function in its first argument

$$f(\alpha;\mathbf{z})=\mathbb{E}_{\mathbf{h}\sim\alpha}[f(\mathbf{h},\mathbf{z})]=\int f(\mathbf{h};\mathbf{z})\alpha[\mathbf{h}]\ .$$

Linearity of the loss and convexity of $\mathcal{M}$ are the key mechanism which allows us to obtain our stronger results. Moreover, if the learning problem is already convex (i.e., f is convex and $\mathcal{H}$ is covex), we can achieve the same results using a deterministic learning rule, as the following claim demonstrates:

Claim 21 *Assume that the hypothesis class $\mathcal{H}$ is convex subset of a vector space, such that $\mathbb{E}_{\mathbf{h}\sim\mathbf{A}(S)}[\mathbf{h}]$ is a well-defined element of $\mathcal{H}$ for any S. Moreover, assume that $f(\mathbf{h};\mathbf{z})$ is convex in $\mathbf{h}$. Then from any (possibly randomized) learning rule $\mathbf{A}$, it is possible to construct a deterministic learning rule $\mathbf{A}'$, such that $f(\mathbf{A}'(S),\mathbf{z})\leq f(\mathbf{A}(S),\mathbf{z})$ for any $S,\mathbf{z}$. As a result, it also holds that $F_S(\mathbf{A}'(S))\leq F_S(\mathbf{A}(S))$ and $F(\mathbf{A}'(S))\leq F(\mathbf{A}(S))$.*

Proof Given a sample S, define $\mathbf{A}'(S;\mathbf{z})$ as the single hypothesis $\mathbb{E}_{\mathbf{h}\sim\mathbf{A}(S)}[\mathbf{h}]$. The proof of the theorem is immediate by Jensen's inequality: since $f()$ is convex in its first argument,

$$f(\mathbf{A}'(S);\mathbf{z})=f(\mathbb{E}_{\mathbf{h}\sim\mathbf{A}(S)}[\mathbf{h}],\mathbf{z})\leq\mathbb{E}_{\mathbf{h}\sim\mathbf{A}(S)}[f(\mathbf{h},\mathbf{z})],$$

where the r.h.s. is in fact $f(\mathbf{A}(S),\mathbf{z})$ by the abuse of notation we have defined previously. ■

Although linearization is the real mechanism at play here, we find it more convenient to display our results and proofs in the language of randomized learning rules.

Allowing randomization allows us to obtain results with respect to the following very strong notion of stability:[4]

Definition 22 *A rule $\mathbf{A}$ is* **strongly-uniform-RO stable** *with rate $\varepsilon_{\text{stable}}(m)$ if for all samples S of m points, for all i, and any $\mathbf{z}',\mathbf{z}'_i\in\mathcal{Z}$, it holds that*

$$\left|f(\mathbf{A}(S^{(i)});\mathbf{z}')-f(\mathbf{A}(S);\mathbf{z}')\right|\leq\varepsilon_{\text{stable}}(m).$$

The strengthening of Theorem 7 that we will prove here is the following:

4. This definition of stability is very similar to the so-called "uniform stability", discussed in Bousquet and Elisseeff (2002), although Bousquet and Elisseeff (2002) consider deterministic learning rules. See Appendix A for more details.

Theorem 23 *A learning problem is learnable if and only if there exists a (possibly randomized) learning rule which is an always AERM and strongly-uniform-RO stable.*

Compared to Theorem 7, we have replaced universal AERM by the stronger notion of an always AERM, and uniform-RO stability by strongly-uniform-RO stability. This makes the result strong enough to formulate a generic learning algorithm, as we will see later on.

The theorem is an immediate consequence of Theorem 7 and the following lemma:

Lemma 24 *For any deterministic learning rule* $\mathbf{A}$*, there exists a randomized learning rule* $\mathbf{A}'$ *such that:*

- *For any* $\mathcal{D}$*, if* $\mathbf{A}$ *is* $\varepsilon_{\text{cons}}$*-consistent under* $\mathcal{D}$ *then* $\mathbf{A}'$ *is* $\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)$ *consistent under* $\mathcal{D}$*.*
- $\mathbf{A}'$ *universally generalizes with rate* $4B/\sqrt{m}$*.*
- *If* $\mathbf{A}$ *is uniform-RO stable with rate* $\varepsilon_{\text{stable}}(m)$*, then* $\mathbf{A}'$ *is strongly-uniform-RO stable with rate* $\varepsilon_{\text{stable}}(\lfloor\sqrt{m}\rfloor)$*.*
- *If* $\mathbf{A}$ *is universally* $\varepsilon_{\text{cons}}$*-consistent, then* $\mathbf{A}'$ *is an always AERM with rate* $2\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)$*.*

Moreover, $\mathbf{A}'$ *is a symmetric learning rule (it does not depend on the order of elements in the sample on which it is applied).*

Proof Consider the learning rule A' which given a sample S, returns a uniform distribution over $A(S')$, where S' ranges over all subsets of S of size $\lfloor\sqrt{m}\rfloor$.

The fact that $\mathbf{A}'$ is symmetric is trivial. We now prove the other assertions in the lemma.

$\mathbf{A}'$ *is consistent:* First note that $F(\mathbf{A}'(S)) = \mathbb{E}_{S'}\left[F(\mathbf{A}(S'))\right]$, and so:

$$\mathbb{E}_S\left[\left|F(\mathbf{A}'(S)) - F^*\right|\right] \;\le\; \mathbb{E}_{S,S'}\left[\left|F(\mathbf{A}(S')) - F^*\right|\right] \;=\; \mathbb{E}_{[S']}\left[\mathbb{E}_{S|[S']}\left[\left|F(\mathbf{A}(S')) - F^*\right|\right]\right]$$

where $[S']$ designates a choice of indices for S'. This decomposition of the random choice of S' (e.g., first deciding on the indices and only then sampling S) allows us think of $[S']$ and S as statistically independent. Given a fixed choice of indices $[S']$, S' is simply an i.i.d. sample of size $\lfloor\sqrt{m}\rfloor$. Therefore, if $\mathbf{A}$ is consistent, $|F(\mathbf{A}(S')) - F^*| \le \varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)$, this holds for any possible fixed $[S']$, and therefore

$$\mathbb{E}_{[S']}\left[\mathbb{E}_{S|[S']}\left[\left|F(\mathbf{A}(S')) - F^*\right|\right]\right] \;=\; \mathbb{E}_{[S']}\left[\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)\right] \;\le\; \varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor).$$

$\mathbf{A}'$ *generalizes:* For convenience, let $b(S,S') = |F_S(\mathbf{A}(S')) - F(\mathbf{A}(S'))|$. Using similar arguments and notation as above:

$$\begin{aligned}
&\mathbb{E}_S\left[\left|F_S(\mathbf{A}'(S)) - F(\mathbf{A}'(S))\right|\right] \\
&\le \mathbb{E}_{[S']}\left[\mathbb{E}_{S|[S']}\left[b(S,S')\right]\right] \\
&\le \mathbb{E}_{[S']}\left[\mathbb{E}_{S|[S']}\left[\frac{\lfloor\sqrt{m}\rfloor}{m} b(S',S')\right] \;+\; \mathbb{E}_{S|[S']}\left[\left(1 - \frac{\lfloor\sqrt{m}\rfloor}{m}\right) b(S\setminus S', S')\right]\right] \\
&\le \mathbb{E}_{[S']}\left[\frac{\lfloor\sqrt{m}\rfloor}{m} 2B + \left(1 - \frac{\lfloor\sqrt{m}\rfloor}{m}\right) \frac{B}{\sqrt{m - \lfloor\sqrt{m}\rfloor} + 1}\right],
\end{aligned}$$

where the last line follows from Lemma 12 and the fact that $b(S,S') \le 2B$ for any S, S'. It is not hard to show that the expression above is at most $4B/\sqrt{m}$, assuming $m \ge 1$.

$\mathbf{A}'$ *is strongly-uniform-RO stable:* For any sample S, any i and replacement instance $\mathbf{z}_i$, and any instance $\mathbf{z}'$, we have that

$$\left|f(\mathbf{A}'(S^{(i)}); \mathbf{z}') - f(\mathbf{A}'(S); \mathbf{z}')\right| \;\le\; \mathbb{E}_{S'}\left[\left|f(\mathbf{A}(S'^{(i)}); \mathbf{z}') - f(\mathbf{A}(S'); \mathbf{z}')\right|\right],$$

where we take $S'^{(i)}$ in the expectation to mean S' if $i \notin [S']$. Notice that if $i \notin [S']$, then $f(\mathbf{A}(S'^{(i)});\mathbf{z}_i) - f(\mathbf{A}(S');\mathbf{z}_i)$ is trivially 0. Thus, we can upper bound the expression above by

$$\mathbb{E}_{S'}\left[\left|f(\mathbf{A}(S'^{i});\mathbf{z}') - f(\mathbf{A}(S');\mathbf{z}')\right| \;\middle|\; i \in [S']\right].$$

Since S' is chosen uniformly over all $\lfloor\sqrt{m}\rfloor$-subsets of S, all permutations of $[S']$ are equally happen to occur, and therefore the above is equal to

$$\mathbb{E}_{S'}\left[\frac{1}{\lfloor\sqrt{m}\rfloor}\sum_{j\in S'}\left|f(\mathbf{A}(S'^{(j)});\mathbf{z}') - f(\mathbf{A}(S');\mathbf{z}')\right|\right] \;\leq\; \mathbb{E}_{S'}\left[\varepsilon_{\text{stable}}(\lfloor\sqrt{m}\rfloor)\right] \;=\; \varepsilon_{\text{stable}}(\lfloor\sqrt{m}\rfloor).$$

$\mathbf{A}'$ *is an always AERM:* For any fixed sample S, we note that

$$\begin{aligned}|F_S(\mathbf{A}'(S)) - F_S(\hat{\mathbf{h}}_S)| \;&=\; \mathbb{E}_{S'}\left[F_S(\mathbf{A}(S')) - F_S(\hat{\mathbf{h}}_S)\right]\\ &= \mathbb{E}_{S'\sim\mathcal{U}(S)^{\lfloor\sqrt{m}\rfloor}}\left[F_S(\mathbf{A}(S')) - F_S(\hat{\mathbf{h}}_S) \mid \text{no dups}\right],\end{aligned}$$

where $\mathcal{U}(S)^{\lfloor\sqrt{m}\rfloor}$ signifies the distribution of i.i.d. samples of size $\lfloor\sqrt{m}\rfloor$, picked uniformly at random (with replacement) from $\lfloor\sqrt{m}\rfloor$, and 'no dups' signifies the event that no element in S was picked twice. By the law of total expectation, this is at most

$$\frac{\mathbb{E}_{S'\sim\mathcal{U}(S)^{\lfloor\sqrt{m}\rfloor}}\left[F_S(\mathbf{A}(S')) - F_S(\hat{\mathbf{h}}_S)\right]}{\mathbb{P}\left[\text{no dups}\right]}.$$

Since the learning rule $\mathbf{A}$ is universally consistent, it is in particular consistent with respect to the distribution $\mathcal{U}(S)$, and therefore the expectation in the expression above is at most $\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)$. As to $\mathbb{P}\left[\text{no dups}\right]$, an analysis identical to the one performed in the proof of Lemma 16 (see Equation (13)) implies that it is at least $1-(\lfloor\sqrt{m}\rfloor)^2/m \geq 1/2$. Overall, we get that $F_S(\mathbf{A}'(S)) - F_S(\hat{\mathbf{h}}_S) \leq 2\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)$, so in particular

$$\frac{\mathbb{E}_{S'\sim\mathcal{U}(S)^{\lfloor\sqrt{m}\rfloor}}\left[F_S(\mathbf{A}(S')) - F_S(\hat{\mathbf{h}}_S)\right]}{\mathbb{P}\left[\text{no dups}\right]} \leq 2\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor),$$

from which the claim follows.

■

6.2 A Generic Learning Algorithm

Recall that a symmetric learning rule $\mathbf{A}$ is such that $\mathbf{A}(S) = \mathbf{A}(S')$ whenever S, S' are identical samples up to permutation. When we deal with randomized learning rules, we assume that the distribution of $\mathbf{A}(S)$ is identical to the distribution of $\mathbf{A}(S')$. Also, let $\bar{\mathcal{H}}$ denote the set of all distributions on $\mathcal{H}$. An element $\bar{\mathbf{h}} \in \bar{H}$ will be thought of as a possible outcome of a randomized learning rule.

Consider the following learning rule: given a sample size m, find a minimizer over all symmetric[5] functions $\mathbf{A} : \mathcal{Z}^m \to \bar{\mathcal{H}}$ of

$$\sup_{S\in\mathcal{Z}^m}\left(F_S(\mathbf{A}(S)) - F_S(\hat{\mathbf{h}}_S)\right) \;+\; \sup_{S\in\mathcal{Z}^m,\mathbf{z}'}\left|f(\mathbf{A}(S);\mathbf{z}') - f(\mathbf{A}(S^{(i)});\mathbf{z}')\right|, \tag{18}$$

with i being an arbitrary fixed element in $\{1,\ldots,m\}$. Once such a function $\mathbf{A}$ is found, return $\mathbf{A}_m(S)$.

5. The algorithm would still work, with slight modifications, if we minimize over all functions - symmetric or not. However, the search space would be larger.

Theorem 25 *If a learning problem is learnable (namely, there exist a universally consistent learning rule with rate $\varepsilon_{\text{cons}}(m)$), the learning algorithm described above is universally consistent with rate*

$$4\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)+\frac{8B}{\sqrt{m}}.$$

Proof By Lemma 24, if a learning problem is learnable, there exists a (possibly randomized) symmetric learning rule $\mathbf{A}'$, which is an always AERM and strongly-uniform-RO stable. More specifically, we have that

$$\sup_{S\in\mathcal{Z}^m}\left(F_S(\mathbf{A}'(S))-F_S(\hat{\mathbf{h}}_S)\right)\leq 2\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor),$$

as well as

$$\sup_{S\in\mathcal{Z}^m,\mathbf{z}'}\left|f(\mathbf{A}'(S);\mathbf{z}')-f(\mathbf{A}'(S^{(i)});\mathbf{z}')\right|\leq\frac{4B}{\sqrt{m}}.$$

In particular, there exists some symmetric $\mathbf{A}:Z^m\to\bar{\mathcal{H}}$, for which the expression in Equation (18) is at most

$$2\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)+\frac{4B}{\sqrt{m}}.$$

Therefore, by definition, the $\mathbf{A}$ found satisfies

$$\sup_{S\in\mathcal{Z}^m}\left(F_S(\mathbf{A}_m(S))-F_S(\hat{\mathbf{h}}_S)\right)\ \leq\ 2\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)+\frac{4B}{\sqrt{m}},\tag{19}$$

as well as

$$\sup_{S\in\mathcal{Z}^m}\left|f(\mathbf{A}_m(S);\mathbf{z}')-f(\mathbf{A}_m(S^{(i)});\mathbf{z}')\right|\ \leq\ 2\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)+\frac{4B}{\sqrt{m}}.\tag{20}$$

In Theorem 9, we have seen that a universally average-RO stable AERM learning rule has to be universally consistent. The inequalities above essentially say that $\mathbf{A}$ is in fact both strongly-uniform-RO stable (and in particular, universally average-RO stable) and an AERM, and thus is a universally consistent learning rule. Formally speaking, this is not entirely accurate, because $\mathbf{A}$ is defined only with respect to samples of size m, and hence is not formally a learning rule which can be applied to samples of any size. However, the analysis we have done earlier in fact carries through also for learning rules $\mathbf{A}$ which are defined just on a specific sample size m. In particular, the analysis of Lemma 11 and Lemma 15 hold verbatim for $\mathbf{A}$ (with trivial modifications due to the fact that $\mathbf{A}$ is randomized), and together imply that since Equation (19) and Equation (20) hold, then

$$\mathbb{E}\left[F(\mathbf{A}(S))-F^*\right]\leq 4\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)+\frac{8B}{\sqrt{m}}.$$

Therefore, our learning algorithm is consistent with rate $4\varepsilon_{\text{cons}}(\lfloor\sqrt{m}\rfloor)+\frac{8B}{\sqrt{m}}$. ■

The main drawback of the algorithm we described is that it is completely infeasible: in practice, we cannot hope to efficiently perform minimization of Equation (18) over all functions from $\mathcal{Z}^m$ to $\bar{\mathcal{H}}$. Nevertheless, we believe it is conceptually important for three reasons: First, it hints that generic methods to develop learning algorithms might be possible in the General Learning Setting (similar to the more specific supervised classification setting); Second, it shows that stability might play a crucial role in the way such methods will work; And third, that stability might act in a similar manner to regularization. Indeed, Equation (18) can be seen as a "regularized ERM" in the space of learning rules (i.e., functions from samples to hypotheses): if we take just the first term in Equation (18), $\sup_{S\in\mathcal{Z}^m}\left(F_S(\mathbf{A}(S))-F_S(\hat{\mathbf{h}}_S)\right)$, then its minimizer is trivially the ERM learning rule. If we take just the second term in Equation (18), $\sup_{S\in\mathcal{Z}^m,\mathbf{z}}\left|f(\mathbf{A}(S);\mathbf{z}')-f(\mathbf{A}(S^{(i)});\mathbf{z}')\right|$, then its minimizers are trivial learning rules which return the same hypothesis irrespective of the training sample. Minimizing a sum of both terms forces us to choose a learning rule which is an "almost"-ERM but also stable - a learning rule which must exist if the problem is learnable at all, as Theorem 23 proves.

In any case, using these results and intuitions to design a generic, *practical* method to learn in the General Learning Setting - remains a very interesting open problem.

7. High Confidence Learnability

So far, we have presented all our results in terms of expectation: namely, the rate at which the expected risk converges to the lowest possible risk. By Markov's inequality, we can always convert these bounds to bounds which hold with probability $1-\delta$ over the sample, and the bounds depend linearly on $1/\delta$. However, in supervised classification, if we have learnability at all, then we have learnability at rates which are logarithmic in $1/\delta$. Can such results be attained in the General Learning Setting?

Fortunately, there is a generic method already known in the literature ("Boosting the Confidence", see Schapire, 1989) which allows us to convert any learning algorithm with linear dependence on δ to an algorithm with logarithmic dependence on $1/\delta$, at a certain price in terms of the sample complexity. This technique is reviewed below.

Moreover, we show that such conversions can in fact be necessary: we give a learning problem which is learnable with an ERM algorithm, and the ERM is stable, but the dependence on the confidence parameter δ cannot be better than linear. This shows that both learnability and stability (under our definitions) of the ERM learning rule are not sufficient to ensure logarithmic dependence on $1/\delta$. Also, this gives a nice illustration to the fundamental differences between the General Learning Setting and supervised classification, where in the latter case learnability implies logarithmic dependence on $1/\delta$.

Theorem 26 *Let* $\mathbf{A}$ *be a universally consistent learning rule with rate* $\varepsilon_{\text{cons}}(m)$*, namely that*

$$\mathbb{E}_{S\sim\mathcal{D}^m}\left[F(\mathbf{A}(S))-F^*\right]\leq\varepsilon_{\text{cons}}(m). \tag{21}$$

Then there exists another universally consistent learning rule $\mathbf{A}'$ *such that with probability at least* $1-\delta$ *over a sample* S *of size* m*,*

$$F(\mathbf{A}'(S))-F^*\leq e\varepsilon_{\text{cons}}\left(\frac{m}{\log(2/\delta)+1}\right)+2B\sqrt{\frac{\log(2/\delta)+\log(\log(2/\delta))}{2m}}$$

Proof Applying Markov's inequality on Equation (21), we have with probability at least $1-1/e$ over a sample S of size m that

$$F(\mathbf{A}(S))-F^*\leq e\varepsilon_{\text{cons}}(m). \tag{22}$$

Now, define the learning rule $\mathbf{A}'$ as follows: given a sample of size m, split it randomly into $a+1$ parts $S_1,\ldots,S_{a+1}$ of size $m/(a+1)$ each (where a is a constant to be determined later). Apply $\mathbf{A}$ separately on $S_1,\ldots,S_a$, to create a hypotheses $\mathbf{A}(S_1),\ldots,\mathbf{A}(S_a)$. Now, return the hypothesis $\mathbf{A}(S_t)$ which minimizes $F_{S_{a+1}}(\mathbf{A}(S_t))$ (namely, the hypothesis with lowest empirical risk on S_{a+1}), where ties are broken arbitrarily. By Equation (22), we have for any S_t separately that with probability at least $1-1/e$,

$$F(\mathbf{A}(S_t))-F^*\leq e\varepsilon_{\text{cons}}\left(\frac{m}{a+1}\right).$$

Since $F(\mathbf{A}(S_1)),\ldots,F(\mathbf{A}(S_a))$ are independent random variables, we have that with probability at least $1-(1/e)^a$, there exists at least one S_t such that

$$F(\mathbf{A}(S_t))-F^*\leq e\varepsilon_{\text{cons}}\left(\frac{m}{a+1}\right).$$

Assume w.l.o.g. that this holds for S_1. Using Hoeffding's inequality and a union bound, it also holds with probability at least $1-\delta_1$ over S that

$$F_{S_{a+1}}(\mathbf{A}(S_1))-F(\mathbf{A}(S_1))\leq B\sqrt{\frac{\log(2a/\delta_1)}{2m}},$$

and also

$$F(\mathbf{A}(S_t))-F_{S_{a+1}}(\mathbf{A}(S_t))\leq B\sqrt{\frac{\log(2a/\delta_1)}{2m}}$$

simultaneously for every $t = 2,\ldots,a$. If this happens, it means that we will pick a hypothesis whose risk is at most $2B\sqrt{\log(2a/\delta_1)/2m}$ larger than $F(\mathbf{A}(S_1))$. Overall, we have that with probability at least $1-\delta_1-(1/e)^a$,

$$F(\mathbf{A}'(S)) - F^* \leq e\varepsilon_{\text{cons}}\left(\frac{m}{a+1}\right) + 2B\sqrt{\frac{\log(2a/\delta_1)}{2m}}\ .$$

Picking $a = \log(2/\delta)$ and $\delta_1 = \delta/2$, we get that with probability at least $1-\delta$,

$$F(\mathbf{A}'(S)) - F^* \leq e\varepsilon_{\text{cons}}\left(\frac{m}{\log(2/\delta)+1}\right) + 2B\sqrt{\frac{\log(4/\delta)+\log(\log(4/\delta))}{2m}}$$

as required. ■

After we have seen how to convert a low-confidence learning rule (linear in δ) to a high-confidence learning rule (logarithmic in δ), we show that such conversions might actually be necessary, in sharp contrast to supervised classification.

Example 3 *There exists a learning problem where any ERM algorithm is universally consistent and average-RO stable with rates $\Theta(1/\sqrt{m})$, but for any ERM algorithm,*

$$\mathbb{P}\left[F(\hat{\mathbf{h}}_S) - F^* = 1\right] = \Theta\left(\frac{1}{\sqrt{m}}\right). \tag{23}$$

The $\Theta(\cdot)$ notation hides only absolute constants.

This example implies that no high-confidence bound is possible, at least without foregoing polynomial dependence on m. To see this, note that a high-confidence result corresponds to $\mathbb{P}\left[F(\hat{\mathbf{h}}_S) - F^*\right] > \varepsilon)$ decreasing exponentially in m for any fixed $\varepsilon > 0$, while in the case above we only have convergence at the rate of $1/\sqrt{m}$.

Proof Consider the instance space $\mathcal{X}\times\mathcal{Y}\times\mathcal{Z} = [0,1]\times\{-1,+1\}\times\{-1,+1\}$, with any joint distribution such that $p(y,z|x)$ for any x is uniform on $\{-1,+1\}^2$, and the marginal distribution on $\mathcal{X}$ is continuous.

Consider the hypothesis class $\mathcal{H} = \mathcal{G}\cup\mathcal{B}$, where $\mathcal{G}$ consists of the constant function 1 and the constant function -1 over $[0,1]$, and $\mathcal{B}$ consists of all functions $\mathbf{h} : [0,1] \mapsto \{-1,0,+1\}$, such that each $\mathbf{h}(\cdot)$ equals 0 on all but a non-empty finite subset of $[0,1]$, and is uniformly either $+1$ or -1 on this finite subset.

Finally, define the objective function as

$$f(\mathbf{h},(x,y,z)) = \left(\mathbf{1}(\mathbf{h}\in\mathcal{G})y + \frac{\mathbf{1}(\mathbf{h}\in\mathcal{B})z}{2|\mathbf{h}|}\right)\mathbf{h}(x) + \mathbf{1}(\mathbf{h}(x)=0),$$

where $|\mathbf{h}| = |\{x\in[0,1] : \mathbf{h}(x)\neq 0\}|$ (namely, the number of points in $[0,1]$ on which the function $h(\cdot)$ is not zero). For $\mathbf{h}\in\mathcal{G}$, where the number of such points is infinite, we take $|\mathbf{h}| = \infty$.

First, notice that for any $\mathbf{h}\in\mathcal{G}$, $F(\mathbf{h}) = 0$, and for any $\mathbf{h}\in B$, $F(\mathbf{h}) = 1$. Thus, we can think of $\mathcal{G}$ as the set of "good" hypotheses, and $\mathcal{B}$ as the set of 'bad" hypotheses. Our goal is to show that any ERM will pick a hypothesis from $\mathcal{B}$ with probability $\Theta(1/\sqrt{m})$.

We need to do a bit of case-by-case analysis. Let $(x_1,y_1,z_1),\ldots,(x_m,y_m,z_m)$ be the sample. If $\sum_{i=1}^m y_i \neq 0$, then using hypotheses in G, it is possible to achieve an empirical risk of

$$-\left|\sum_{i=1}^m y_i\right| \leq -1,$$

while using hypotheses in B, it is only possible to achieve an empirical risk of

$$\frac{\sum_{i=1}^m z_i\mathbf{h}(x_i)}{2|\mathbf{h}|} + \sum_{i=1}^m \mathbf{1}(\mathbf{h}(x_i)=0) \geq -\frac{1}{2}.$$

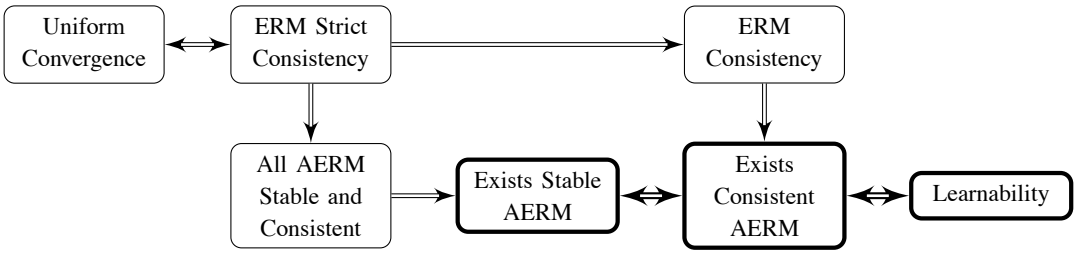


Figure 2: Implications of various properties of learning problems. Consistency refers to univeral consistency and stability refers to universal uniform-RO stability.

Thus, with probability $1-\Theta(1/\sqrt{m})$ (the probability that $\sum_{i=1}^{m} y_i \neq 0$ in the sample), any ERM algorithm will pick $\hat{\mathbf{h}}_S \in G$.

If $\sum_{i=1}^{m} y_i = 0$, then any $\mathbf{h} \in G$ achieves an empirical objective value of 0. On the other hand, unless $\sum_{i=1}^{m} z_i = 0$, we can choose some $\mathbf{h} \in B$, which is non-zero on all points in the sample, and achieves an empirical risk smaller than 0. The probability that $\sum_{i=1}^{m} y_i = 0$ and $\sum_{i=1}^{m} z_i \neq 0$ is $\Theta(1/\sqrt{m})(1-\Theta(1/\sqrt{m}))$, or $\Theta(1/\sqrt{m})$.

So we have that any ERM picks $\hat{\mathbf{h}}_S \in G$ with probability $1-\Theta(1/\sqrt{m})$, and some $\hat{\mathbf{h}}_S \in B$ with probability $\Theta(1/\sqrt{m})$, from which the consistency rate and Equation (23) in the theorem statement follows. Finally, note that replacing a single instance in the training set will lead to the ERM picking a different hypothesis, only if $\sum_{i=1}^{m} y_i = 0$ before or after the replacement. The probability for getting a training set where this happens is $O(1/\sqrt{m})$, and from this it is easy to see that the ERM is average-RO stable with rate $O(1/\sqrt{m})$. ■

8. Discussion and Conclusions

In the familiar setting of supervised classification problems, the question of learnability is reduced to that of uniform convergence of empirical risks to their expectations. Therefore, for the purposes of establishing learnability, there is no need to look beyond the ERM.

In this paper, we showed that in the General Learning Setting, which includes more general problems, this equivalence does not hold, and the situation is substantially more complex. ERM might work without any uniform convergence, and learnability might be possible only with a non-ERM algorithm. We are therefore in need of a new understanding of the question of learnability, that applies more broadly than just to supervised classification.

In studying learnability in the General Setting, Vapnik (1995) focuses solely on empirical risk minimization, which we have seen to be insufficient for understanding learnability. Furthermore, for empirical risk minimization, Vapnik establishes uniform convergence as a necessary and sufficient condition not for ERM consistency, but rather for *strict* consistency of the ERM. We have seen that even in rather non-trivial problems, where the ERM is consistent and generalizes, strict consistency does not hold. This perhaps gives an indication that strict consistency might be too strict.

On the other hand, we have seen that stability is both a sufficient and necessary condition for learning, even in the General Learning Setting where uniform convergence fails to characterize learnability. A previous stability-based characterization (Mukherjee et al., 2006) relied on uniform convergence and thus applied only to restricted setting. Extending the characterization beyond these settings is particularly interesting, since for supervised classification the question of learnability is already essentially solved. This also allows us to frame stability as the core condition guaranteeing learnability, with uniform convergence only a sufficient, but not necessary, condition for stability (see Figure 2).

In studying the question of learnability and its relation to stability, we encountered several differences between this more general setting, and settings such as supervised classification where learnability is equivalent to uniform convergence. We summarize some of these distinctions:

- Perhaps the most important distinction is that in the General Setting learnability might be possible only with a non-ERM. In this paper we establish that if a problem is learnable, although it might not be learnable with an ERM, it must be learnable with some AERM. And so, in the General Setting we must look beyond empirical risk minimization, but not beyond asymptotic empirical risk minimization.

- In supervised classification, if one AERM is universally consistent then all AERMs are universally consistent. In the General Setting we must choose the AERM carefully.

- In supervised classification, a universally consistent rule must also generalize and be AERM. In the General Setting, a universally consistent rule need not generalize nor be an AERM, as example 2 demonstrates. However, Theorem 10 establishes that, even in the General Setting, if a rule is universally consistent *and* generalizing then it must be an AERM. This gives us another reason to not look beyond asymptotic empirical risk minimization, even in the General Setting.

  The above distinctions can also be seen through Corollary 19, which is concerned with the relationship between AERM, consistency and generalization in learnable problems. In the General Setting, any two conditions imply the other, but it is possible for any one condition to exist without the others. In supervised classification, if a problem is learnable then generalization always holds (for any rule), and so universal consistency and AERM imply each other.
- In supervised classification, ERM inconsistency for some distribution is enough to establish non-learnability. Establishing non-learnability in the General Setting is trickier, since one must consider all AERMs. We show how Corollary 19 can provide a *certificate* for non-learnability, in the form of a rule that is consistent and an AERM for some specific distribution, but does not generalize (Example 1).

- In supervised classification, any learnable problem is learnable with an ERM, *and* the ERM "works" with high-confidence (namely, $F(\hat{\mathbf{h}}_S) - F^*$ can be bounded with probability $1-\delta$ by an expression with logarithmic dependence on $1/\delta$). In Section 7 we have seen that in the General Learning Setting, even if the ERM is universally consistent, high-confidence bounds for the ERM might be impossible to obtain.

We have begun exploring the issue of learnability in the General Setting, and uncovered important relationships between learnability and stability. But many problems are left open, some of which are listed below.

First, is it possible to come up with well-known machine learning applications, where learnability is achievable despite uniform convergence failing to hold?

In Section 6.2, we have managed to obtain a completely generic learning algorithm: an algorithm which in principle allows us to learn any learnable problem. However, the algorithm suffers from the severe drawback that in general, it requires unbounded computational power. Can we derive an efficient algorithm, or characterize classes of learning problems where our algorithm, or some other generic learning algorithm using the notion of stability, can be executed efficiently? For instance, can we always learn using a regularized ERM learning rule?

On a related vein, it would be interesting to develop learning algorithms (perhaps for specific settings rather than generic learning problems) which directly use stability in order to learn. Convex regularization is one such mechanism, as discussed in Section 4. Are there other mechanisms, which use the notion of stability in a different way?

Another issue is that even the existence of uniform-RO stable AERM (or strongly-uniform-RO stable, always-AERM allowing for convexity/randomization) is not as elegant and simple as having finite VC dimension or fat-shattering dimension. It would be very interesting to derive equivalent but more "combinatorial" conditions for learnability.

Yet another open question: We showed that existence of an uniform-RO stable AERM is necessary and sufficient for learnability (Theorem 7). However, it is possible that learnability is an equivalent to the existence of an AERM with a stronger notion of stability, without resorting to convexity/randomization as we

have done in Section 6.2. This might perhaps lead to generic learning algorithms which perform minimization over a search space more feasible than the one our algorithm (in Section 6.2) uses.

Finally, we do not know whether it is enough to consider symmetric learning rules: that is, learning rules which do not depend on the order of the instances in the training sample. Intuitively, this should be true, since the instances were sampled i.i.d. Can our characterization of learnability (e.g., existence of a uniform-RO stable AERM learning rule) be strengthened to existence of symmetric uniform-RO stable AERM learning rule, without allowing convexity/randomization?

Acknowledgments

We would like to thank Leon Bottou, Vladimir Vapnik and Tong Zhang for helpful discussions. We would also like to thank the anonymous reviewers for their detailed and helpful comments.

Appendix A. Alternative Notions of Stability

In this appendix, we discuss how our definition of stability compares to previous definitions in the literature, as well as demonstrate how subtleties involved in the precise choice of the definition can have a significant effect on the results which can be obtained.

A.1 Previous Definitions in the Literature

The existing literature on stability in learning, briefly surveyed in Section 3.2, uses many different stability measures. All of them measure the amount of change in the algorithm's output as a function of small changes to the sample on which the algorithm is run. However, they differ in how "output", "amount of change to the output", and "small changes to the sample" are defined. In Section 5, we used three stability measures. Roughly speaking, one measure (average-RO stability) is the expected change in the objective value on a particular instance, after that instance is replaced with a different instance. The second measure and third measure (uniform-RO stability and strongly-uniform-RO stability respectively) basically deal with the maximal possible change in the objective value with respect to a particular instance, by replacing a single instance in the training set. However, instead of measuring the objective value on a specific instance, we could have measured the change in the risk of the returned hypothesis, or any other distance measure between hypotheses. Instead of replacing an instance, we could have talked about adding or removing one instance from the sample, either in expectation or in some arbitrary manner. Such variations are common in the literature.

To relate our stability definitions to the ones in the literature, we note that our definitions of uniform-RO stability and strongly-uniform-RO stability are somewhat similar to uniform stability (Bousquet and Elisseeff, 2002), which in our notation is defined as $\sup_{S,\mathbf{z}} \max_i |f(\mathbf{A}(S;\mathbf{z})) - f(\mathbf{A}(S^{\setminus i});\mathbf{z})|$, where $S^{\setminus i}$ is the training sample S with instance $\mathbf{z}_i$ removed. Compared to uniform-RO stability, here we measure maximal change over any particular instance, rather than average change over all instances in the training sample. Also, we deal with removing an instance rather than replacing it. Strongly-uniform-RO stability is more similar, with the only formal difference being removal vs. replacement of an instance. However, the results for uniform stability mostly assume deterministic learning rules, while in this paper we have used strongly-uniform-RO stability solely in the context of randomized learning rules. For deterministic learning rules, the differences outlined above are sufficient to make uniform stability a strictly stronger requirement than uniform-RO stability, since it is easy to come up with learning problems and (non-symmetric) learning rules which are uniform-RO stable but not uniformly stable. Moreover, we show in this paper that uniform-RO stable AERM's characterize learnability, while it is well known that uniformly stable AERM's are not necessary for learnability (see Kutin and Niyogi, 2002). For the same reason, our notion of strongly-uniform-RO stability is apparently too strong to characterize learnability when we deal with deterministic learning rules, as opposed to randomized learning rules.

Our definition of average-RO stable is similar to "average stability" defined in Rakhlin et al. (2005), which in our notation is defined as $\mathbb{E}_{S\sim\mathcal{D}^m,\mathbf{z}'_1}\left[f(\mathbf{A}(S^{(i)});\mathbf{z}_1)-f(\mathbf{A}(S);\mathbf{z}_1)\right]$. Compared to average-RO stability, the main difference is that the change in the objective value is measured with respect to $\mathbf{z}_1$ rather than an average over $\mathbf{z}_i$ for all i, and stems from the assumption there that the learning algorithm is symmetric. Notice that in this paper we do not make such an assumption.

For an elaborate study on other stability notions and their relationships, see Kutin and Niyogi (2002).

Unfortunately, many of the stability notions in the literature are incomparable, and even slight changes in the definition radically affect their behavior. We go into this in much more detail in the following subsections.

A.2 LOO Stability vs. RO Stability

The stability definitions we have used in this paper are all based on the idea of replacing one instance in the training sample by another instance (e.g., "RO" or "replace-one" stability). An alternative set of definitions can be obtained based on *removing* one instance in the training sample (e.g., "LOO" or "leave-one-out" stability). In fact, these were the definitions used in our preliminary paper (Shalev-Shwartz et al., 2009b). Despite seeming like a small change, it turns out there is a considerable discrepancy in terms of the obtainable results, compared to RO stability. In this subsection, we wish to discuss these discrepancies, as well as show how small changes to the stability definition can materially affect its strength.

Specifically, we consider the following four LOO stability measures, each slightly weaker than the previous one. The first and last are similar to our notion of uniform-RO stability and average-RO stability respectively. However, we emphasize that RO stability and LOO stability are in general incomparable notions, as we shall see later on. Also, we note that some of these definitions appeared in previous literature. For instance, the notion of "all-i-LOO" below has been studied by several authors under different names (Bousquet and Elisseeff, 2002; Mukherjee et al., 2006; Rakhlin et al., 2005). The notation $S^{\setminus i}$ below refer to a training sample S with instance $\mathbf{z}_i$ removed.

Definition 27 *A rule* $\mathbf{A}$ *is* **uniform-LOO stable** *with rate* $\varepsilon_{\text{stable}}(m)$ *if for all samples* S *of* m *points and for all* i*:*

$$\left|f(\mathbf{A}(S^{\setminus i});\mathbf{z}_i)-f(\mathbf{A}(S);\mathbf{z}_i)\right|\leq\varepsilon_{\text{stable}}(m).$$

Definition 28 *A rule* $\mathbf{A}$ *is* **all-i-LOO stable** *with rate* $\varepsilon_{\text{stable}}(m)$ *under distribution* $\mathcal{D}$ *if for all* i*:*

$$\mathbb{E}_{S\sim\mathcal{D}^m}\left[\left|f(\mathbf{A}(S^{\setminus i});\mathbf{z}_i)-f(\mathbf{A}(S);\mathbf{z}_i)\right|\right]\leq\varepsilon_{\text{stable}}(m).$$

Definition 29 *A rule* $\mathbf{A}$ *is* **LOO stable** *with rate* $\varepsilon_{\text{stable}}(m)$ *under distribution* $\mathcal{D}$ *if*

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{S\sim\mathcal{D}^m}\left[\left|f(\mathbf{A}(S^{\setminus i});\mathbf{z}_i)-f(\mathbf{A}(S);\mathbf{z}_i)\right|\right]\leq\varepsilon_{\text{stable}}(m).$$

Definition 30 *A rule* $\mathbf{A}$ *is* **on-average-LOO stable** *with rate* $\varepsilon_{\text{stable}}(m)$ *under distribution* $\mathcal{D}$ *if*

$$\left|\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{S\sim\mathcal{D}^m}\left[f(\mathbf{A}(S^{\setminus i});\mathbf{z}_i)-f(\mathbf{A}(S);\mathbf{z}_i)\right]\right|\leq\varepsilon_{\text{stable}}(m).$$

While some of the definitions above might look rather similar, we show below that each one is strictly stronger than the other. Example 6 is interesting in its own right, since it presents a learning problem and an AERM that is universally consistent, but not LOO stable. While this is possible in the General Learning Setting, in supervised classification every such AERM has to be LOO stable (this is essentially proven in Mukherjee et al., 2006).

Example 4 *There exists a learning problem with a universally consistent and all-i-LOO stable learning rule, but there is no universally consistent and uniform LOO stable learning rule.*

Proof This example is taken from Kutin and Niyogi (2002). Consider the hypothesis space $\{0,1\}$, the instance space $\{0,1\}$, and the objective function $f(\mathbf{h},z) = |h-z|$.

It is straightforward to verify that an ERM is a universally consistent learning rule. It is also universally all-i-LOO stable, because removing an instance can change the hypothesis only if the original sample had an equal number of 0's and 1's (plus or minus one), which happens with probability at most $O(1/\sqrt{m})$ where m is the sample size. However, it is not hard to see that the only uniform LOO stable learning rule, at least for large enough sample sizes, is a constant rule which always returns the same hypothesis h regardless of the sample. Such a learning rule is obviously not universally consistent. ■

Example 5 *There exists a learning problem with a universally consistent and LOO-stable AERM, which is not symmetric and is not all-i-LOO stable.*

Proof Let the instance space be $[0,1]$, the hypothesis space $[0,1] \cup 2$, and the objective function $f(h,z) = \mathbb{1}_{\{h=z\}}$. Consider the following learning rule $\mathbf{A}$: given a sample, check if the value z_1 appears more than once in the sample. If no, return z_1, otherwise return 2.

Since $F_S(2) = 0$, and z_1 returns only if this value constitutes $1/m$ of the sample, the rule above is an AERM with rate $\varepsilon_{\text{erm}}(m) = 1/m$. To see universal consistency, let $\mathbb{P}[z_1] = p$. With probability $(1-p)^{m-2}$, $z_1 \notin \{z_2,\ldots,z_m\}$, and the returned hypothesis is z_1, with $F(z_1) = p$. Otherwise, the returned hypothesis is 2, with $F(2) = 0$. Hence $\mathbb{E}_S[F(\mathbf{A}(S))] \leq p(1-p)^{m-2}$, which can be easily verified to be at most $1/(m-1)$, so the learning rule is consistent with rate $\varepsilon_{\text{cons}}(m) \leq 1/(m-1)$. To see LOO-stability, notice that our learning hypothesis can change by deleting z_i, $i > 1$, only if z_i is the only instance in $z_2,\ldots,z_m$ equal to z_1. So $\varepsilon_{\text{stable}}(m) \leq 2/m$ (in fact, LOO-stability holds even without the expectation). However, this learning rule is not all-i-LOO-stable. For instance, for any continuous distribution, $|f(\mathbf{A}(S^{\setminus 1}),z_1) - f(\mathbf{A}(S),z_1)| = 1$ with probability 1, so it obviously cannot be all-i-LOO-stable with respect to $i = 1$. ■

Example 6 *There exists a learning problem with a universally consistent (and on-average-LOO stable) AERM, which is not LOO stable.*

Proof Let the instance space, hypothesis space and objective function be as in Example 4. Consider the following learning rule, based on a sample $S = (z_1,\ldots,z_m)$: if $\sum_i \mathbb{1}_{\{z_i=1\}}/m > 1/2 + \sqrt{\log(4)/2m}$, return 1. If $\sum_i \mathbb{1}_{\{z_i=1\}}/m < 1/2 - \sqrt{\log(4)/2m}$, return 0. Otherwise, return $\text{Parity}(S) = (z_1 + \ldots z_m) \bmod 2$.

This learning rule is an AERM, with $\varepsilon_{\text{erm}}(m) = \sqrt{2\log(4)/m}$. Since we have only two hypotheses, we have uniform convergence of $F_S(\cdot)$ to $F(\cdot)$ for any hypothesis. Therefore, our learning rule universally generalizes (with rate $\varepsilon_{\text{gen}}(m) = \sqrt{\log(4/\delta)/2m}$), and by Theorem 9, this implies that the learning rule is also universally consistent and on-average-LOO stable.

However, the learning rule is not LOO stable. Consider the uniform distribution on the instance space. By Hoeffding's inequality, $|\sum_i \mathbb{1}_{\{z_i=1\}}/m - 1/2| \leq \sqrt{\log(4)/2m}$ with probability at least $1/2$ for any sample size m. In that case, the returned hypothesis is the parity function (even when we remove an instance from the sample, assuming $m \geq 3$). When this happens, it is not hard to see that for any i,

$$f(\mathbf{A}(S),z_i) - f(\mathbf{A}(S^{\setminus i}),z_i) = \mathbb{1}_{\{z_i=1\}}(-1)^{\text{Parity(S)}}.$$

This implies that

$$\begin{aligned}
&\mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}\left|\left(f(\mathbf{A}(S^{\setminus i});\mathbf{z}_i)-f(\mathbf{A}(S);\mathbf{z}_i)\right)\right|\right] \qquad (24)\\
&\geq \frac{1}{2}\mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}_{\{z_i=1\}}\left|\sqrt{\frac{\log(4)}{2m}}\geq\left|\sum_{i=1}^{m}\frac{\mathbb{1}_{\{z_i=1\}}}{m}-\frac{1}{2}\right|\right|\right]\\
&\geq \frac{1}{2}\left(\frac{1}{2}-\sqrt{\frac{\log(4)}{2m}}\right) \;\longrightarrow\; \frac{1}{4}\ ,
\end{aligned}$$

which does not converge to zero with the sample size m. Therefore, the learning rule is not LOO stable. ■

Note that the proof implies that on-average-LOO stability cannot be replaced even by something between on-average-LOO stability and LOO stability. For instance, a natural candidate would be

$$\mathbb{E}_{S\sim\mathcal{D}^m}\left[\left|\frac{1}{m}\sum_{i=1}^{m}\left(f(\mathbf{A}(S^{\setminus i});\mathbf{z}_i)-f(\mathbf{A}(S);\mathbf{z}_i)\right)\right|\right], \qquad (25)$$

where the absolute value is now over the entire sum, but inside the expectation. In the example used in the proof, Equation (25) is still lower bounded by Equation (24), which does not converge to zero with the sample size.

After showing that the hierarchy of definitions above is indeed strict, we turn to the question of what can be characterized in terms of LOO stability. In Shalev-Shwartz et al. (2009b), we show a version of Theorem 7, which asserts that a problem is learnable if and only if there is an on-average-LOO stable AERM. However, on-average-LOO stability is qualitatively much weaker than the notion of uniform-RO stability used in Theorem 7 (see Definition 4). Rather, we would expect to prove a version of the theorem with the notion of uniform-LOO stability or at least LOO stability, which are more analogous to uniform-RO stability. However, the proof of Theorem 7 does not work for these stability definitions (technically, this is because the proof relies on the sample size remaining constant, which is true for replacement stability, but not when we remove an instance as in LOO stability). We do not know if one can prove a version of Theorem 7 with an LOO stability notion stronger than on-average-LOO stability.

On the plus side, LOO stability allows us to prove the following interesting result, specific to ERM learning rules.

Theorem 31 *For an ERM the following are equivalent:*

- *Universal LOO stability.*
- *Universal consistency.*
- *Universal generalization.*

In particular, the theorem implies that LOO stability is a necessary property for consistent ERM learning rules. This parallels Theorem 9, which dealt with AERM's in general, and used RO stability. As before, we do not know how to obtain something akin to Theorem 9 with RO stability.

Proof Lemma 15 and Lemma 17 from Section 5.3.3 already tell us that for ERM's, universal consistency is equivalent to universal generalization. Moreover, Lemma 14 implies that for ERM's, generalization is equivalent to on-average generalization (see Equation (11) for the exact definition). Thus, is left to prove that for ERM's, generalization implies LOO stability, and LOO stability implies on-average generalization. stability.

First, suppose the ERM learning rule is generalizing with rate $\varepsilon_{\text{gen}}(m)$. Note that $f(\hat{\mathbf{h}}_{S^{\setminus i}}; z_i) - f(\hat{\mathbf{h}}_S; z_i)$ is always nonnegative. Therefore the LOO stability of the ERM can be upper bounded as follows:

$$\begin{aligned}
&\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[\left|f(\hat{\mathbf{h}}_{S^{\setminus i}}; z_i) - f(\hat{\mathbf{h}}_S; z_i)\right|\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[f(\hat{\mathbf{h}}_{S^{\setminus i}}; z_i) - f(\hat{\mathbf{h}}_S; z_i)\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[F(\hat{\mathbf{h}}_{S^{\setminus i}})\right] - \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} f(\hat{\mathbf{h}}_S; z_i)\right] \\
&\leq \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[F_{S^{\setminus i}}(\hat{\mathbf{h}}_{S^{\setminus i}}) + \varepsilon_{\text{gen}}(m-1)\right] - \mathbb{E}\left[F_S(\hat{\mathbf{h}}_S)\right] \\
&= \varepsilon_{\text{gen}}(m-1) + \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} F_{S^{\setminus i}}(\hat{\mathbf{h}}_{S^{\setminus i}}) - F_S(\hat{\mathbf{h}}_S)\right] \\
&\leq \varepsilon_{\text{gen}}(m-1).
\end{aligned}$$

For the opposite direction, suppose the ERM learning rule is LOO stable with rate $\varepsilon_{\text{stable}}(m)$. Notice that we can get any sample of size $m-1$ by picking a sample S of size m and discarding any instance i. Therefore, the on-average generalization rate of the ERM for samples of size $m-1$ is equal to the following:

$$\begin{aligned}
&\left|\mathbb{E}\left[F(\hat{\mathbf{h}}_{S^{\setminus i}}) - F_{S^{\setminus i}}(\hat{\mathbf{h}}_{S^{\setminus i}})\right]\right| \\
&= \left|\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[F(\hat{\mathbf{h}}_{S^{\setminus i}}) - F_{S^{\setminus i}}(\hat{\mathbf{h}}_{S^{\setminus i}})\right]\right| \\
&= \left|\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[f(\hat{\mathbf{h}}_{S^{\setminus i}}; z_i)\right] - \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[F_{S^{\setminus i}}(\hat{\mathbf{h}}_{S^{\setminus i}})\right]\right|
\end{aligned}$$

Now, note that for the ERM's of S and $S^{\setminus i}$ we have $\left|F_{S^{\setminus i}}(\hat{\mathbf{h}}_{S^{\setminus i}}) - F_S(\hat{\mathbf{h}}_S)\right| \leq \frac{2B}{m}$. Therefore, we can upper bound the above by

$$\begin{aligned}
&\left|\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[f(\hat{\mathbf{h}}_{S^{\setminus i}}; z_i)\right] - \mathbb{E}\left[F_S(\hat{\mathbf{h}}_S)\right]\right| + \frac{2B}{m} \\
&= \left|\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[f(\hat{\mathbf{h}}_{S^{\setminus i}}; z_i) - f(\hat{\mathbf{h}}_{S,}; z_i)\right]\right| \\
&\leq \varepsilon_{\text{stable}}(m)
\end{aligned}$$

using the assumption that the learning rule is $\varepsilon_{\text{stable}}(m)$-stable. ■

References

N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM*, 44(4):615–631, 1997.

M. Anthony and P. Bartlet. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965, October 1989.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002. ISSN 1533-7928.

L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California at Berkeley, 1996.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, September 2004.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, 13:49–52, 1902.

D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

M. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11(6):1427–1453, 1999.

M. Kearns, R. Schapire, and L. Sellie. Toward efficient agnostic learning. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 341–352, 1992.

S. Kutin and P. Niyogi. Almost-everywhere algorithmic stability and generalization error. In *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence*, pages 275–282, 2002.

D. McAllester and R. Schapire. On the convergence rate of Good-Turing estimators. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 1–6, 2000.

S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25(1-3):161–193, 2006.

D. L. Phillips. A technique for the numerical solution of certain integral equations of the first kind. *Journal of the ACM*, 9(1):84–97, 1962.

S. Rakhlin, S. Mukherjee, and T. Poggio. Stability results in learning theory. *Analysis and Applications*, 3 (4):397–419, 2005.

W. Rogers and T. Wagner. A finite sample distribution-free performance bound for local discrimination rules. *Annals of Statistics*, 6(3):506–514, 1978.

R.E. Schapire. The strength of weak learnability. In *30th Annual Symposium on Foundations of Computer Science*, pages 28–33, October 1989.

S. Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University, 2007.

S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Stochastic convex optimization. In *Proceedings of the 22nd Annual Conference on Computational Learning Theory*, 2009a.

S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Learnability and stability in the general learning setting. In *Proceedings of the 22nd Annual Conference on Computational Learning Theory*, 2009b.

K. Sridharan, N. Srebro, and S. Shalev-Shwartz. Fast rates for regularized objectives. In *Advances in Neural Information Processing Systems 22*, pages 1545–1552, 2008.

A. N. Tikhonov. On the stability of inverse problems. *Dolk. Akad. Nauk SSSR*, 39(5):195–198, 1943.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, XVI(2):264–280, 1971.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 928–936, 2003.

Journal of Machine Learning Research 11 (2010) 2671-2705 Submitted 10/09; Revised 5/10; Published 10/10

Topology Selection in Graphical Models of Autoregressive Processes

Jitkomut Songsiri JITKOMUT@EE.UCLA.EDU
Lieven Vandenberghe VANDENBE@EE.UCLA.EDU
Department of Electrical Engineering
University of California
Berkeley, CA 90095-1594, USA

Editor: Martin Wainwright

Abstract

An algorithm is presented for topology selection in graphical models of autoregressive Gaussian time series. The graph topology of the model represents the sparsity pattern of the inverse spectrum of the time series and characterizes conditional independence relations between the variables. The method proposed in the paper is based on an ℓ_1-type nonsmooth regularization of the conditional maximum likelihood estimation problem. We show that this reduces to a convex optimization problem and describe a large-scale algorithm that solves the dual problem via the gradient projection method. Results of experiments with randomly generated and real data sets are also included.

Keywords: graphical models, time series, topology selection, convex optimization

1. Introduction

We consider graphical models of autoregressive (AR) Gaussian processes

$$x(t) = -\sum_{k=1}^{p} A_k x(t-k) + w(t), \qquad w(t) \sim \mathcal{N}(0,\Sigma) \tag{1}$$

where $x(t) \in \mathbf{R}^n$, and $w(t) \in \mathbf{R}^n$ is Gaussian white noise. A graphical model of the time series is an undirected graph with n nodes, one for each component $x_i(t)$, and an edge connecting nodes i and j if the components $x_i(t)$ and $x_j(t)$ are *conditionally dependent*, given the other components of the time series. The conditional independence property has a simple characterization (which holds for general Gaussian stationary processes) in terms of the spectrum of the process: $x_i(t)$ and $x_j(t)$ are independent, conditional on the other $n-2$ components of $x(t)$, if and only if

$$(S(\omega)^{-1})_{ij} = 0$$

for all ω, where $S(\omega)$ is the spectral density matrix (Brillinger, 1981, Chapter 8; Dahlhaus, 2000). This characterization allows us to include the conditional independence relations in an estimation problem by placing sparsity constraints on the inverse spectral density matrix.

In Songsiri et al. (2009) a convex optimization method was discussed for estimating the model parameters A_k, Σ from data, given the graph of conditional independence relations. The method is

based on solving the convex optimization problem

$$\begin{array}{ll} \text{minimize} & -\log\det X_{00} + \mathbf{tr}(CX) \\ \text{subject to} & Y_k = \sum_{i=0}^{p-k} X_{i,i+k}, \quad k = 0,1,\ldots,p, \\ & (Y_k)_{ij} = 0, \quad (i,j) \in \mathcal{V}, \quad k = 0,1,\ldots,p, \\ & X \succeq 0. \end{array} \tag{2}$$

Here C is the sample covariance matrix and $\mathcal{V}$ is the set of conditionally independent pairs of variables. The optimization variables are $X \in \mathbf{S}^{n(p+1)}$ (the symmetric matrices of order $n(p+1)$), $Y_0 \in \mathbf{S}^n$, and $Y_k \in \mathbf{R}^{n\times n}$, $k = 1,2,\ldots,p$. X_{ij} denotes the $n \times n$ subblock of X in position i, j, where the indices i and j run from 0 to p. It was shown that if the sample covariance matrix C is block-Toeplitz, then problem (2) is equivalent to the conditional maximum likelihood (ML) estimation problem, and the ML estimates for A_k and Σ are easily obtained from the optimal solution X. If C is not block-Toeplitz, the problem is a relaxation and in general not equivalent to the conditional ML problem. However in practice, the relaxation often happens to be exact (Songsiri et al., 2009). This will be discussed in more detail in §2.3.

In this paper we consider the more general problem of estimating the model parameters *and* the topology of the graphical model. The topology selection problem can be solved by enumerating all topologies, solving the ML estimation problem for each topology, and ranking them via information-theoretic criteria such as the Akaike or Bayes information criteria (Eichler, 2006; Songsiri et al., 2009). However this combinatorial approach is clearly limited to small graphs. The goal of this paper is to present an efficient alternative based on convex optimization.

Topology selection for graphical models of time series is of interest in many applications (see Dahlhaus et al., 1997; Eichler et al., 2003; Salvador et al., 2005; Gather et al., 2002; Timmer et al., 2000; Feiler et al., 2005; Friedman et al., 2008). A common approach is to formulate hypothesis testing problems to decide about the presence or absence of edges. Dahlhaus (2000) derives a statistical test for the existence of an edge in the graph, based on the maximum of a nonparametric estimate of the normalized inverse spectrum (see also Dahlhaus et al., 1997; Eichler et al., 2003; Salvador et al., 2005; Gather et al., 2002; Timmer et al., 2000; Feiler et al., 2005; Fried and Didelez, 2003). Eichler (2008) presents a more general approach by introducing a hypothesis test based on the norm of some suitable function of the spectral density matrix. A related problem was studied by Bach and Jordan (2004). They use an efficient search procedure to learn the graph structure from sample estimates of the joint spectral density matrix.

If $p = 0$, the problem (2) reduces to

$$\begin{array}{ll} \text{minimize} & -\log\det X + \mathbf{tr}(CX) \\ \text{subject to} & X_{ij} = 0, \quad (i,j) \in \mathcal{V} \end{array} \tag{3}$$

with variable $X \in \mathbf{S}^n$. (Throughout the paper we take the set of positive definite matrices as the domain of the function $\log\det X$, so (3) includes an implicit constraint $X \succ 0$.) Problem (3) is known as the *covariance selection* problem, that is, the problem of computing the ML estimate of the inverse covariance matrix $X = \Sigma^{-1}$ of a multivariate Gaussian variable $\mathcal{N}(0,\Sigma)$, subject to conditional independence constraints (which, for a normal distribution, correspond to zeros in the inverse covariance); see Dempster (1972) and Lauritzen (1996, §5.2). Recently, new heuristic methods for topology selection in large Gaussian graphical models have been developed. These methods are

based on augmenting the ML objective with an ℓ_1-norm regularization term, that is, on solving

$$\text{minimize} \quad -\log\det X + \mathbf{tr}(CX) + \gamma \textstyle\sum_{ij} |X_{ij}| \tag{4}$$

(see Dahl et al., 2005; Meinshausen and Bühlmann, 2006; Banerjee et al., 2008; Ravikumar et al., 2008; Friedman et al., 2008; Lu, 2009, 2010). The optimization problem (4) is convex but has $n(n+1)/2$ variables (the elements of X) and is nondifferentiable, so it can be challenging to solve when n is large. Several large-scale methods have been proposed. Banerjee et al. (2008) apply a block coordinate descent method to the dual problem. Each step of this method reduces to solving a quadratic program with box constraints. They also apply Nesterov's optimal gradient method (Nesterov, 2005) to a smooth approximation of (4). Friedman et al. (2008) observe that the dual of the subproblems in the coordinate descent algorithm can be regarded as a lasso-type problem and solved with a method called graphical lasso. Scheinberg and Rish (2009) consider a coordinate ascent method applied to the primal problem. A method based on column-wise updates is given by Rothman et al. (2008). A related problem is explored in Yuan and Lin (2007) where the authors make a connection between (4) and more general determinant maximization problems (Vandenberghe et al., 1998), and solve the problem using interior-point methods. Lu (2009) observes that the dual of (4) is a smooth problem and applies Nesterov's method (Nesterov, 2005) directly to the dual. The algorithm is further extended by Lu (2010) and compared with a projected spectral gradient method. Another closely related paper is Duchi et al. (2008) in which the gradient projection method is applied to the dual problem.

The main purpose of this paper is to develop an efficient method for topology selection in AR models, based on augmenting the estimation problem (2) with a convex regularization term, similar to the ℓ_1-norm regularization used in (4). We also discuss first-order methods for solving the resulting large-scale and nondifferentiable convex optimization problem.

The paper is organized as follows. In Section 2 we review the definition of conditional independence in time series and summarize the results from Songsiri et al. (2009). In Section 3 we set up the topology selection problem as a regularized ML problem and discuss its properties. Examples and applications are presented in Sections 4 and 5. We conclude in Section 6 with a discussion of gradient projection algorithms for solving large instances of the regularized ML estimation problem.

1.1 Notation

$\mathbf{S}^n$ is the set of real symmetric matrices of order n. $\mathbf{S}^n_+$ and $\mathbf{S}^n_{++}$ are the sets of symmetric positive semidefinite, respectively, positive definite, matrices of order n. $\mathbf{R}^{m\times n}$ is the set of $m\times n$-matrices. $\mathbf{M}^{n,p}$ is the set of matrices

$$X = \begin{bmatrix} X_0 & X_1 & \cdots & X_p \end{bmatrix}$$

with $X_0 \in \mathbf{S}^n$ and $X_1,\ldots,X_p \in \mathbf{R}^{n\times n}$. The standard trace inner product $\langle X,Y\rangle = \mathbf{tr}(X^TY)$ is used for the three vector spaces $\mathbf{S}^n, \mathbf{R}^{m\times n}, \mathbf{M}^{n,p}$. For a symmetric matrix X, the inequalities $X \succeq 0$ and $X \succ 0$ mean X is positive semidefinite, resp., positive definite. Row and column indices of submatrices in a block matrix start at 0. If X is a matrix with (block) entries X_{ij}, then $X_{i:j,k:l}$ will denote the submatrix formed by rows i through j and columns k through l:

$$X_{i:j,k:l} = \begin{bmatrix} X_{ik} & X_{i,k+1} & \cdots & X_{il} \\ X_{i+1,k} & X_{i+1,k+1} & \cdots & X_{i+1,l} \\ \vdots & \vdots & & \vdots \\ X_{jk} & X_{j,k+1} & \cdots & X_{j,l} \end{bmatrix}.$$

The linear mapping $\mathrm{T} : \mathbf{M}^{n,p} \to \mathbf{S}^{n(p+1)}$ constructs a symmetric block Toeplitz matrix from its first block row: if $X = \begin{bmatrix} X_0 & X_1 & \cdots & X_p \end{bmatrix} \in \mathbf{M}^{n,p}$, then

$$\mathrm{T}(X) = \begin{bmatrix} X_0 & X_1 & \cdots & X_p \\ X_1^T & X_0 & \cdots & X_{p-1} \\ \vdots & \vdots & \ddots & \vdots \\ X_p^T & X_{p-1}^T & \cdots & X_0 \end{bmatrix}. \tag{5}$$

The adjoint of T is a mapping $\mathrm{D} : \mathbf{S}^{n(p+1)} \to \mathbf{M}^{n,p}$ defined as follows. If $S \in \mathbf{S}^{n(p+1)}$ is partitioned as

$$S = \begin{bmatrix} S_{00} & S_{01} & \cdots & S_{0p} \\ S_{01}^T & S_{11} & \cdots & S_{1p} \\ \vdots & \vdots & & \vdots \\ S_{0p}^T & S_{1p}^T & \cdots & S_{pp} \end{bmatrix},$$

then $\mathrm{D}(S) = \begin{bmatrix} \mathrm{D}_0(S) & \mathrm{D}_1(S) & \cdots & \mathrm{D}_p(S) \end{bmatrix}$ where

$$\mathrm{D}_0(S) = \sum_{i=0}^{p} S_{ii}, \qquad \mathrm{D}_k(S) = 2\sum_{i=0}^{p-k} S_{i,i+k}, \quad k = 1, \ldots, p. \tag{6}$$

A symmetric sparsity pattern of a sparse matrix X of order n will be associated with the positions $\mathcal{V} \subseteq \{1,\ldots,n\} \times \{1,\ldots,n\}$ of its zero entries. We assume $(i,i) \notin \mathcal{V}$ for $i = 1,\ldots,n$, that is, the diagonal entries are not included among the zeros. $\mathrm{P}_{\mathcal{V}}(X)$ denotes the projection of a matrix $X \in \mathbf{S}^n$ or $X \in \mathbf{R}^{n\times n}$ on the complement of the sparsity pattern $\mathcal{V}$:

$$\mathrm{P}_{\mathcal{V}}(X)_{ij} = \begin{cases} X_{ij} & (i,j) \in \mathcal{V} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The same notation is used for $\mathrm{P}_{\mathcal{V}}$ as a mapping from $\mathbf{R}^{n\times n} \to \mathbf{R}^{n\times n}$ and as a mapping from $\mathbf{S}^n \to \mathbf{S}^n$. In both cases, $\mathrm{P}_{\mathcal{V}}$ is self-adjoint. If X is an $r \times s$ block matrix with i,j block X_{ij}, and each block is square of order n, then $\mathrm{P}_{\mathcal{V}}(X)$ denotes the $r \times s$ block matrix with i,j block $\mathrm{P}_{\mathcal{V}}(X)_{ij} = \mathrm{P}_{\mathcal{V}}(X_{ij})$. Similarly, $\mathrm{P}_{\mathcal{V}}(X)$ with $X \in \mathbf{M}^{n,p}$ denotes

$$\begin{bmatrix} \mathrm{P}_{\mathcal{V}}(X_0) & \mathrm{P}_{\mathcal{V}}(X_1) & \cdots & \mathrm{P}_{\mathcal{V}}(X_p) \end{bmatrix}.$$

The subscript $\mathcal{V}$ in $\mathrm{P}_{\mathcal{V}}$ is omitted if the sparsity pattern is clear from the context.

2. Graphical Models of Autoregressive Gaussian Processes

In this section we describe the conditional independence property for Gaussian time series, review the maximum likelihood estimation of AR models, and provide a convex formulation for the estimation problem with conditional independence constraints.

2.1 Conditional Independence

Let $x(t)$ be an n-dimensional stationary zero-mean Gaussian process with spectrum $S(\omega)$:

$$S(\omega) = \sum_{k=-\infty}^{\infty} R_k e^{-\mathrm{j}k\omega}$$

where $R_k = \mathbf{E}x(t+k)x(t)^T$ and $\mathrm{j} = \sqrt{-1}$. We assume that $S(\omega)$ is invertible for all ω. Two components $x_i(t)$ and $x_j(t)$ of $x(t)$ are conditionally independent (i.e., conditional on the other components of $x(t)$) if

$$(S(\omega)^{-1})_{ij} = 0$$

for all ω (Brillinger, 1981; Dahlhaus, 2000). If we denote by $\mathcal{V}$ the set of index pairs i, j of conditionally independent variables, then we can use the projection operator $\mathrm{P} = \mathrm{P}_{\mathcal{V}}$ defined in (7) to express the conditional independence relations as

$$\mathrm{P}(S(\omega)^{-1}) = 0. \tag{8}$$

In a graphical model of the process, the index set $\mathcal{V}$ is the set of missing edges in the graph.

To apply this result to AR processes (1) we need to express the inverse spectrum in terms of the model parameters. The notation will simplify if we first normalize the input covariance and use the model

$$B_0 x(t) = -\sum_{k=1}^{p} B_k x(t-k) + v(t), \qquad v(t) \sim \mathcal{N}(0, I), \tag{9}$$

where $B_0 \in \mathbf{S}^n_{++}$ and $B_k \in \mathbf{R}^{n\times n}$, $k = 1, \ldots, p$. If Σ is nonsingular, the two models are equivalent, and related as $B_0 = \Sigma^{-1/2}$, $B_k = \Sigma^{-1/2} A_k$ for $k \geq 1$. The inverse spectrum $S(\omega)$ of the process (9) is a trigonometric matrix polynomial

$$S(\omega)^{-1} = Y_0 + \frac{1}{2}\sum_{k=1}^{p} (e^{-\mathrm{j}k\omega} Y_k + e^{\mathrm{j}k\omega} Y_k^T) \tag{10}$$

where $Y_0 = \sum_{l=0}^{p} B_l^T B_l$, and $Y_k = 2\sum_{l=0}^{p-k} B_l^T B_{k+l}$ for $k = 1, \ldots, p$. If we define $B = \begin{bmatrix} B_0 & B_1 & \cdots & B_p \end{bmatrix}$, we can use the operator D defined in (6) to express Y_k as

$$\begin{bmatrix} Y_0 & Y_1 & \cdots & Y_p \end{bmatrix} = \mathrm{D}(B^T B).$$

The expression (10) shows that $(S(\omega)^{-1})_{ij}$ is identically zero if and only if the i, j and j, i entries of Y_k are zero for $k = 0, \ldots, p$. The conditional independence condition (8) is therefore equivalent to a quadratic equation in the model parameters B_k:

$$\mathrm{P}\left(\mathrm{D}(B^T B)\right) = 0. \tag{11}$$

(Recall from the Notation section that if Y is a block matrix with square submatrices Y_k of order n, then $\mathrm{P}(Y)$ denotes the block matrix with submatrices $\mathrm{P}(Y_k)$.)

2.2 Conditional Maximum Likelihood Estimation

We now consider the problem of estimating the model parameters B from an observed sequence $\tilde{x}(1), \tilde{x}(2), \ldots, \tilde{x}(N)$ of the AR process, subject to known conditional independence constraints (11). In Songsiri et al. (2009) the estimation problem was formulated as the optimization problem

$$\begin{array}{ll} \text{minimize} & -2\log\det B_0 + \mathbf{tr}(CB^T B) \\ \text{subject to} & \mathrm{P}(\mathrm{D}(B^T B)) = 0. \end{array} \tag{12}$$

The matrix $C \in \mathbf{S}_+^{n(p+1)}$ is a sample estimate of the covariance matrix, that is, its blocks C_{ij}, $i \leq j$, are estimates of the covariances $R_{j-i} = \mathbf{E}x(t+j-i)x(t)^T$, calculated from the observed sequence. Two choices of C are common. The first choice is the *non-windowed estimate*

$$C = \frac{1}{N-p}HH^T, \qquad H = \begin{bmatrix} \tilde{x}(p+1) & \tilde{x}(p+2) & \cdots & \tilde{x}(N) \\ \tilde{x}(p) & \tilde{x}(p+1) & \cdots & \tilde{x}(N-1) \\ \vdots & \vdots & & \vdots \\ \tilde{x}(1) & \tilde{x}(2) & \cdots & \tilde{x}(N-p) \end{bmatrix}. \tag{13}$$

With this choice the estimation problem (12) can be interpreted as a maximum likelihood problem. Indeed, from (9), the conditional density of a sequence $x(t_1), x(t_1+1), \ldots x(t_2)$, given $x(t_1-p), \ldots, x(t_1-1)$, is given by

$$\left(\frac{\det B_0}{(2\pi)^{n/2}}\right)^{t_2-t_1+1} \exp\left(-\frac{1}{2}\sum_{t=t_1}^{t_2} \mathbf{x}(t)^T B^T B \mathbf{x}(t)\right),$$

where $\mathbf{x}(t)$ denotes the $n(p+1)$-vector $\mathbf{x}(t) = (x(t), x(t-1), \ldots, x(t-p))$. From this it can be shown that the cost function in (12) with C defined as in (13), is essentially the negative conditional log-likelihood function of the observed sequence $\tilde{x}(p+1)$, $\tilde{x}(p+2)$, $\ldots$, $\tilde{x}(N)$, given $\tilde{x}(1)$, $\ldots$, $\tilde{x}(p)$. We therefore refer to (12) as the *conditional maximum likelihood problem*. For AR processes, the conditional ML formulation is substantially simpler and more often used than the exact ML formulation. Moreover, when the data length N is sufficiently large compared to p, the difference between the exact and conditional ML formulations is small.

The second choice for C is the *windowed estimate*

$$C = \frac{1}{N}HH^T, \tag{14}$$

where

$$H = \begin{bmatrix} \tilde{x}(1) & \tilde{x}(2) & \cdots & \tilde{x}(p+1) & \cdots & \tilde{x}(N) & 0 & \cdots & 0 \\ 0 & \tilde{x}(1) & \cdots & \tilde{x}(p) & \cdots & \tilde{x}(N-1) & \tilde{x}(N) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{x}(1) & \cdots & \tilde{x}(N-p) & \tilde{x}(N-p+1) & \cdots & \tilde{x}(N) \end{bmatrix}.$$

The windowed estimate C is block-Toeplitz, and this guarantees several useful properties of the resulting model B (for example, stability; see Songsiri et al., 2009). In practice, the differences between the windowed and non-windowed estimates are small when $N \gg p$.

We will assume that C is positive definite. If n is small compared to N, this is a reasonable assumption but not guaranteed to be true. (As a counterexample, assume $\tilde{x}(1)$, $\ldots$, $\tilde{x}(n)$ are the first n unit vectors and the remainder of the sequence is zero. The matrix C in (14) then has rank $n+p$.) If C is not positive definite, it may be necessary to add a small multiple of the identity. This is equivalent to a quadratic regularization term proportional to $\|B\|_F^2$ in the objective of (12).

When there are no sparsity constraints in (12), the solution can be found by setting the gradient of the cost function equal to zero, which gives

$$\begin{bmatrix} C_{00} & C_{01} & \cdots & C_{0p} \\ C_{10} & C_{11} & \cdots & C_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ C_{p0} & C_{p1} & \cdots & C_{pp} \end{bmatrix} \begin{bmatrix} B_0 \\ B_1^T \\ \vdots \\ B_p^T \end{bmatrix} = \begin{bmatrix} B_0^{-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Written in terms of the original variables $\Sigma = B_0^{-2}$, $A_k = B_0^{-1}B_k$, this gives

$$\begin{bmatrix} C_{00} & C_{01} & \cdots & C_{0p} \\ C_{10} & C_{11} & \cdots & C_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ C_{p0} & C_{p1} & \cdots & C_{pp} \end{bmatrix} \begin{bmatrix} I \\ A_1^T \\ \vdots \\ A_p^T \end{bmatrix} = \begin{bmatrix} \Sigma \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{15}$$

with unknowns $\Sigma = B_0^{-2}$, $A_k = B_0^{-1}B_k$. The bottom p equations form a set of linear equations from which $A_1, \ldots, A_p$ can be determined. Plugging in the solution in the first equation gives Σ. Later in the paper we will refer to the solution as the *least-squares estimate* because the bottom p equations can be interpreted as normal equations for the least-squares problem

$$\text{minimize} \quad \mathbf{tr}(ACA^T)$$

with variable $A = \begin{bmatrix} I & A_1 & \cdots & A_p \end{bmatrix}$. This method is also known as the *covariance method* if C is the non-windowed sample covariance (13), and as the *correlation method* if C is the windowed sample covariance (14) (see Stoica and Moses, 1997).

2.3 Convex Formulation

The optimization problem (12) is non-convex because of the quadratic equality constraint. A convex relaxation is

$$\begin{array}{ll} \text{minimize} & -\log\det X_{00} + \mathbf{tr}(CX) \\ \text{subject to} & \mathrm{P}(\mathrm{D}(X)) = 0 \\ & X \succeq 0 \end{array} \tag{16}$$

with variable $X \in \mathbf{S}^{n(p+1)}$. The relaxation is exact, that is, the two problems (16) and (12) are equivalent, if the optimal solution X of (16) has rank n. In that case, the solution B of (16) can be calculated by factoring X as $X = B^T B$.

A condition for exactness of the relaxation follows from the dual problem of (16), which is

$$\begin{array}{ll} \text{maximize} & \log\det W + n \\ \text{subject to} & \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix} \preceq C + \mathrm{T}(\mathrm{P}(Z)), \end{array} \tag{17}$$

with variables $W \in \mathbf{S}^n$ and $Z \in \mathbf{M}^{n,p}$ (for the derivation, see Songsiri et al., 2009). The variable Z is the Lagrange multiplier associated with the equality constraint in (16); the slack matrix in the inequality in (17) is the multiplier associated with the primal inequality $X \succeq 0$. To find the relation between primal and dual solutions, we first note that the primal and dual problems are strictly feasible: $X = I$ is strictly feasible in the primal problem (16), since by assumption $\mathcal{V}$ does not contain any diagonal entries; in the dual problem $Z = 0$ and a sufficiently small positive definite W are strictly feasible, because $C \succ 0$ by assumption. From convex duality, strict primal and dual feasibility imply that the primal and dual problems are solvable, and that their optimal solutions are related by the optimality conditions

$$X_{00}^{-1} = W, \qquad \mathbf{tr}\left(X\left(C + \mathrm{T}(\mathrm{P}(Z)) - \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix}\right)\right) = 0 \tag{18}$$

(Boyd and Vandenberghe, 2004, Chapter 5). The second condition is known as *complementary slackness* between the optimal X and the dual variable associated with the inequality $X \succeq 0$. From these optimality conditions, it can be shown that the relaxation is exact when the trailing principal submatrix of order np in the matrix $C + \mathrm{T}(\mathrm{P}(Z)) \in \mathbf{S}^{n(p+1)}$ is positive definite at the optimum, that is,

$$(C + \mathrm{T}(\mathrm{P}(Z)))_{1:p,1:p} \succ 0. \tag{19}$$

Under this condition, the rank of

$$C + \mathrm{T}(\mathrm{P}(Z)) - \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix}$$

is at least np. Since X has order $n(p+1)$, the two conditions in (18) imply that the optimal X has rank n.

In general it is difficult to guarantee a priori that the condition (19) holds at optimum. However, when C is block-Toeplitz, then (19) can be shown to hold for all dual feasible Z. This follows from the following easily established property of block-Toeplitz matrices: if $V \in \mathbf{S}^{n(p+1)}$ is a symmetric block-Toeplitz matrix with $n \times n$ blocks V_{ij}, and

$$V = \begin{bmatrix} V_{00} & V_{0,1:p} \\ V_{1:p,0} & V_{1:p,1:p} \end{bmatrix} \succeq \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix}$$

for some $W \succ 0$, then V is positive definite (see Songsiri et al., 2009, §3.3.3). We therefore conclude that for positive definite block-Toeplitz C (for example, the windowed sample covariance (14) or the true covariance), the problems (12) and (16) are *equivalent*. For general non-block-Toeplitz C (for example, the non-windowed sample covariance (13)), we cannot guarantee that (19) holds at the optimum. However, we can note that the non-windowed sample covariance approaches a block-Toeplitz matrix as $N \to \infty$. It is therefore not surprising that even for the non-windowed estimate, the relaxation is often exact, as was observed in the experimental results in Songsiri et al. (2009).

3. Topology Selection Via Nonsmooth Regularization

In the previous section we have described a convex formulation of the (conditional) ML estimation problem with given conditional independence constraints, that is, a given graph topology. In many applications the topology is not known, and needs to be discovered from the data. Information theoretic model selection criteria such as the Akaike, second-order Akaike, or Bayes information criteria can be used for this purpose. They require enumerating all possible topologies, solving the ML problem for each topology, and ranking the ML estimates according to their information criterion score. These scores are defined as

$$\mathrm{AIC} = -2\mathcal{L} + 2k, \qquad \mathrm{AIC_c} = -2\mathcal{L} + \frac{2Nk}{N-k-1}, \qquad \mathrm{BIC} = -2\mathcal{L} + k\log N \tag{20}$$

where $\mathcal{L}$ is the log-likelihood of the ML estimate, N is the sample size, and k is the effective number of parameters. In our application, $\mathcal{L}$ is given by

$$\mathcal{L} = \frac{N-p}{2}\left(\log\det X_{00} - \mathbf{tr}(CX)\right)$$

where X is the optimal solution of (16), and we use for k the total number of parameters in the estimation problem,

$$k = \frac{n(n+1)}{2} - |\mathcal{V}| + p(n^2 - 2|\mathcal{V}|),$$

where $|\mathcal{V}|$ is the number of conditionally independent pairs of variables. This topology selection method based on information-theoretic criteria is feasible if the number of possible topologies is not too large, but quickly becomes intractable even for small values of n. In this section and the next we describe a more scalable approach based on a convex optimization problem that extends the ℓ_1-norm heuristic (4) for sparse covariance selection.

3.1 Regularized ML Problem

In analogy with the convex heuristic for covariance selection (4), we can formulate a regularized ML problem by adding a nonsmooth ℓ_1-type penalty:

$$\begin{array}{ll} \text{minimize} & -\log\det X_{00} + \mathbf{tr}(CX) + \gamma h(\mathrm{D}(X)) \\ \text{subject to} & X \succeq 0, \end{array} \tag{21}$$

where $\gamma > 0$ is a weighting parameter. The penalty $h : \mathbf{M}^{n,p} \to \mathbf{R}$ is a convex function, chosen to encourage a sparse solution X with a common, symmetric sparsity pattern for the $p+1$ blocks of $\mathrm{D}(X)$. We will use the penalty function

$$h_\infty(Y) = \sum_{j>i} \max\left\{ \left|(Y_0)_{ij}\right|, \max_{k=1,\ldots,p} \left|(Y_k)_{ij}\right|, \max_{k=1,\ldots,p} \left|(Y_k)_{ji}\right| \right\} \tag{22}$$

that is, a sum of the ℓ_∞-norms of vectors of i,j and j,i-entries of the coefficients Y_k. In the examples (Section 4) we will also discuss penalty functions defined as sums of ℓ_α-norms, with $\alpha = 1, 2$.

Regularization with a convex sum-of-norms penalty is a popular technique for achieving sparsity of groups of variables. Examples from statistics are the *composite absolute penalties* (CAP) (Zhao et al., 2009) and the *group lasso* (Yuan and Lin, 2006; Kim et al., 2006). When $p = 0$ and $X \in \mathbf{S}^n$ in (21) the penalty term reduces to $\sum_{i>j} |X_{ij}|$ and we obtain problem (4), studied in Banerjee et al. (2008), Lu (2009) and Friedman et al. (2008), with the minor difference that we do not penalize the diagonal entries of X.

We now derive the dual problem of (21) which will be important in Section 6. To simplify the derivation we introduce a variable $Y = \mathrm{D}(X)$ and write the problem as

$$\begin{array}{ll} \text{minimize} & -\log\det X_{00} + \mathbf{tr}(CX) + \gamma h_\infty(Y) \\ \text{subject to} & Y = \mathrm{D}(X) \\ & X \succeq 0. \end{array}$$

If we use a multiplier $Z \in \mathbf{M}^{n,p}$ for the equality constraint $Y = \mathrm{D}(X)$ and a multiplier $U \in \mathbf{S}^{n(p+1)}$ for the inequality $X \succeq 0$, the Lagrangian of the problem is

$$\begin{aligned} L(X,Y,Z,U) &= -\log\det X_{00} + \mathbf{tr}(CX) + \gamma h_\infty(Y) - \mathbf{tr}(UX) + \mathbf{tr}(Z^T(\mathrm{D}(X) - Y)) \\ &= -\log\det X_{00} + \mathbf{tr}((C + \mathrm{T}(Z) - U)X) + \gamma h_\infty(Y) - \mathbf{tr}(Z^T Y). \end{aligned} \tag{23}$$

(Recall that the mappings T and D defined in (5) and (6) are adjoints, that is, $\mathbf{tr}(Z^T \mathrm{D}(X)) = \mathbf{tr}(\mathrm{T}(Z)X)$.) The dual function is the infimum of the Lagrangian over X and Y. We first minimize over Y. The nonlinear penalty term does not depend on the diagonal entries of the blocks Y_k. The minimization over the diagonal entries of Y_k is therefore unbounded below unless

$$\mathbf{diag}(Z_k) = 0, \quad k = 0, 1, \ldots, p. \tag{24}$$

The minimization over the off-diagonal part of the blocks Y_k decomposes into independent minimizations of the functions

$$-\sum_{k=0}^{p} ((Z_k)_{ij}(Y_k)_{ij} + (Z_k)_{ji}(Y_k)_{ji}) + \gamma \max\left\{ |(Y_0)_{ij}|, \max_{k=1,\ldots,p} |(Y_k)_{ij}|, \max_{k=1,\ldots,p} |(Y_k)_{ji}| \right\}$$

for each element i, j with $i > j$. This expression is unbounded below unless

$$2|(Z_0)_{ij}| + \sum_{k=1}^{p} (|(Z_k)_{ij}| + |(Z_k)_{ji}|) \leq \gamma, \quad i \neq j, \tag{25}$$

and, if this condition holds, the infimum over Y is zero. The result of the partial minimization of the Lagrangian over Y can be summarized as

$$\inf_Y L(X,Y,Z,U) = \begin{cases} -\log\det X_{00} + \mathbf{tr}((C + \mathrm{T}(Z) - U)X) & (24), (25) \\ -\infty & \text{otherwise.} \end{cases}$$

Next, we carry out the minimization over X. The terms in X_{00} are bounded below if and only if $(C + \mathrm{T}(Z) - U)_{00} \succ 0$, and if this holds, they are minimized by $X_{00} = (C + \mathrm{T}(Z) - U)_{00}^{-1}$. The Lagrangian is linear in the other blocks X_{ij}, and therefore bounded below (and identically zero) only if $(C + \mathrm{T}(Z) - U)_{ij} = 0$ for blocks $(i, j) \neq (0, 0)$. This gives a third set of dual feasibility conditions:

$$(C + \mathrm{T}(Z) - U)_{00} \succ 0, \qquad (C + \mathrm{T}(Z) - U)_{ij} = 0, \quad (i, j) \neq 0, \tag{26}$$

and an expression for the dual function

$$g(Z,U) = \inf_{X,Y} L(X,Y,Z,U) = \begin{cases} \log\det(C + \mathrm{T}(Z) - U)_{00} + n & (24), (25), (26) \\ -\infty & \text{otherwise.} \end{cases}$$

The dual problem is to maximize $g(Z,U)$ subject to $U \succeq 0$. If we add a variable $W = C_{00} + Z_0 - U_{00}$ and eliminate the slack variable U, we can express the dual problem as

$$\begin{array}{ll} \text{maximize} & \log\det W + n \\ \text{subject to} & \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix} \preceq C + \mathrm{T}(Z) \\ & \sum_{k=0}^{p} (|(Z_k)_{ij}| + |(Z_k)_{ji}|) \leq \gamma, \quad i \neq j \\ & \mathbf{diag}(Z_k) = 0, \quad k = 0, \ldots, p. \end{array} \tag{27}$$

The variables are $W \in \mathbf{S}^n$ and $Z \in \mathbf{M}^{n,p}$. When $p = 0$, the problem reduces to

$$\begin{array}{ll} \text{maximize} & \log\det(C + Z) + n \\ \text{subject to} & |Z_{ij}| \leq \gamma/2, \quad i \neq j \\ & \mathbf{diag}(Z) = 0, \end{array}$$

Except for the equality constraint, this is the problem considered in Lu (2009) and Duchi et al. (2008).

If a sum of ℓ_α-norms

$$h_\alpha(Y) = \sum_{j>i} \left(\sum_{k=0}^{p} (|(Y_k)_{ij}|^\alpha + |(Y_k)_{ji}|^\alpha) \right)^{1/\alpha} \tag{28}$$

is used as penalty function in (21), the second constraint in the corresponding dual problem (27) is replaced by

$$\left(\sum_{k=0}^{p} \left(|(Z_k)_{ij}|^\beta + |(Z_k)_{ji}|^\beta \right) \right)^{1/\beta} \leq \gamma, \quad i \neq j$$

with $\beta = \alpha/(\alpha-1)$.

3.2 Optimality Conditions

The primal problem (21) is always strictly feasible ($X = I$ is strictly feasible). The dual problem (21) is strictly feasible if $C \succ 0$ (we can take $Z = 0$ and W positive definite and sufficiently small). It follows that the primal and dual problems are solvable, have equal optimal values, and that their solutions are characterized by the following set of necessary and sufficient optimality (or KKT) conditions.

Primal feasibility. X and Y satisfy

$$X \succeq 0, \qquad X_{00} \succ 0, \qquad Y = \mathrm{D}(X).$$

Dual feasibility. W and Z satisfy

$$W \succ 0, \qquad C + \mathrm{T}(Z) \succeq \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix},$$

$$\sum_{k=0}^{p} (|(Z_k)_{ij}| + |(Z_k)_{ji}|) \leq \gamma, \quad i \neq j, \qquad \mathbf{diag}(Z_k) = 0, \quad k = 0, 1, \ldots p.$$

Zero duality gap. The Lagrangian evaluated at the primal and dual optimal solutions is equal to the primal objective at the optimal X, Y, and equal to the dual objective evaluated at the optimal W, Z. From (23), we have equality between the Lagrangian and the primal objective if $\mathbf{tr}(UX) = 0$. Therefore the complementary slackness condition

$$\mathbf{tr}\left(X \left(C + \mathrm{T}(Z) - \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix} \right) \right) = 0 \tag{29}$$

holds at the optimum. Equality between the Lagrangian and the dual objective requires that the primal optimal X, Y minimize the Lagrangian evaluated at the dual optimal W, Z. Reviewing the derivation of the dual problem, we see that X_{00} minimizes the Lagrangian if

$$X_{00}^{-1} = W. \tag{30}$$

To express the conditions from the minimization over Y, we define

$$t_{ij} = \max\left\{ |(Y_0)_{ij}|, \max_{k=1,\ldots,p} |(Y_k)_{ij}|, \max_{k=1,\ldots,p} |(Y_k)_{ji}| \right\}.$$

Then we see that Y minimizes the Lagrangian if for all $i \neq j$, we either have

$$\sum_{k=0}^{p} (|(Z_k)_{ij}| + |(Z_k)_{ji}|) < \gamma,$$

or we have $\sum_{k=0}^{p}(|(Z_k)_{ij}| + |(Z_k)_{ji}|) = \gamma$ and

$$(Z_k)_{ij} = 0,\ |(Y_k)_{ij}| \leq t_{ij} \quad \text{or} \quad (Z_k)_{ij} < 0,\ (Y_k)_{ij} = -t_{ij} \quad \text{or} \quad (Z_k)_{ij} > 0,\ (Y_k)_{ij} = t_{ij}$$

for $k = 0, \ldots, p$.

The conditions (29)–(30) show that the optimal X has rank n under the same conditions as for the problem with given sparsity pattern (16). If

$$(C + \mathrm{T}(Z))_{1:p,1:p} \succ 0$$

then the optimal X has rank n, and this is always the case if C is block-Toeplitz. Under these conditions, the optimization problem (21) is equivalent to a regularized (conditional) ML estimation problem for the model parameters B:

$$\text{minimize} \quad -2\log\det B_0 + \mathbf{tr}(CB^TB) + \gamma h_\infty(\mathrm{D}(B^TB)).$$

4. Examples with Randomly Generated Data

Our interest in the regularized ML formulation (21) is motivated by the fact that the resulting AR model typically has a sparse inverse spectrum $S(\omega)^{-1}$. Since the regularized problem is convex, it is interesting as an efficient heuristic for topology selection. In this section we illustrate several aspects of this approach using experiments with randomly generated data. In Section 5 we will apply the method to real data sets. Numerical algorithms for solving the regularized problem (21) are discussed in Section 6.

4.1 Method

We first explain in greater detail how we will use the results of the regularized ML problem for model selection.

4.1.1 CHOICE OF REGULARIZATION PARAMETER γ

The sparsity in the inverse spectrum of the solution of the regularized ML problem is controlled by the weighting coefficient γ. As γ varies, the sparsity pattern varies from dense (γ small) to diagonal (γ large). Several authors have discussed the choice of γ in the context of covariance selection (i.e., heuristics based on solving problem (4) or closely related problems). A common approach is to select γ via cross-validation; see, for example, Friedman et al. (2008), Huang et al. (2006) and Banerjee et al. (2008). Meinshausen and Bühlmann (2006) give explicit formulas for γ based

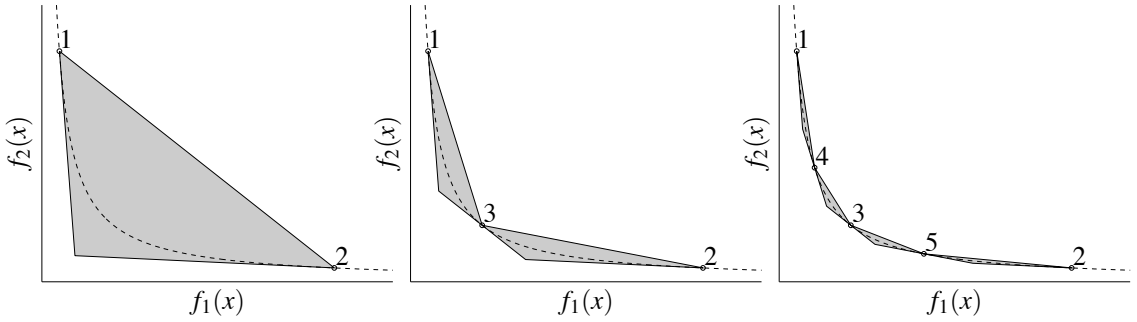


Figure 1: Method for approximating the trade-off curve between two convex objectives.

on a statistical analysis of the probability of errors in the topology (see also Yuan and Lin, 2007; Banerjee et al., 2008). Asadi et al. (2009) consider γ as a random variable and use a maximum a posteriori probability (MAP) estimation to choose γ and the covariance matrix.

In the examples of this section we will use the following method for selecting γ. We first compute the entire trade-off curve between the two terms in the objective of (21), that is, between the log-likelihood and the penalty function $h_\infty(\mathrm{D}(X))$. The trade-off curve can be computed by solving (21) for a number of different values of γ (see below). We collect the topologies of the solutions along the trade-off curve, and solve the ML problem (16) for each of these topologies. We then rank the models using the Bayes information criterion (BIC), as discussed at the beginning of Section 3, and select the model with the lowest score. In this approach, the convex heuristic is used as a preprocessing step to reduce the number of topologies that are examined using the BIC, and to filter out topologies that are unlikely to be competitive.

4.1.2 TRACING TRADE-OFF CURVES

The trade-off curves are computed by solving (21) for a sequence of values of γ. To obtain an accurate estimate of the curve with only a small number of values γ we use a method which is illustrated in Figure 1 for a generic trade-off between two convex cost functions f_1 and f_2. We first solve the scalarized problem

$$\text{minimize} \quad f_1(x) + \gamma f_2(x) \tag{31}$$

for two positive values γ_1, γ_2 near the opposite ends of the trade-off curve. This gives the points labeled 1 and 2 on the trade-off curve. The values of γ_1 and γ_2 also define the slopes of straight lines that support the trade-off curve at points 1 and 2. Since the trade-off curve is convex, we can conclude that the curve between 1 and 2 lies somewhere in the shaded triangular region. As γ_3, we choose the value that corresponds to the slope of the straight line between 1 and 2. Solving problem (31) with $\gamma = \gamma_3$ gives point 3 on the trade-off curve and a straight line that supports the curve at point 3. The trade-off curve between points 1 and 2 is now known to lie in the union of the two shaded triangles. Next, we solve the problem (31) for a value γ_4 corresponding to the slope of the straight line between points 1 and 3, and a value γ_5 corresponding to the slope of the straight line between 3 and 2. In this example, we obtain fairly accurate upper and lower bounds of the actual trade-off curve after solving five scalarized problems (31).

4.1.3 THRESHOLDING

With a proper value of γ, the regularized ML problem (21) has a sparse solution Y, resulting in a sparse inverse spectrum $S(\omega)^{-1}$. When solved with a limited accuracy, the entries of Y are not exactly zero. We will use the following method to determine the topology from the computed solution.

We calculate the inverse spectrum $S(\omega)^{-1}$ and normalize it by scaling its rows and columns so that the diagonal is one:

$$R(\omega) = \mathbf{diag}(S(\omega)^{-1})^{-1/2} S(\omega)^{-1} \mathbf{diag}(S(\omega)^{-1})^{-1/2}.$$

The normalized inverse spectrum $R(\omega)$ is known as the *partial coherence* (Brillinger, 1981; Dahlhaus, 2000). Its entries are between 0 and 1 in magnitude, and measure the conditional dependence between the corresponding variables, after removing the linear effects from the other variables. In the static case ($p = 0$), $R(\omega)$ reduces to the normalized concentration matrix. To estimate the graph topology we compare the L_∞-norms of the entries of $R(\omega)$,

$$\rho_{ij} = \sup_{\omega} |R(\omega)_{ij}|$$

with a given threshold. This thresholding step is similar to thresholding in other sparse methods, for example the thresholded lasso and Dantzig estimators in Lounici (2008).

To simplify the interpretation we will use the same threshold value (10^{-1}) in all the experiments, that is, we remove edge (i, j) from the graph if $\rho_{ij} \leq 10^{-1}$.

4.2 Experiment 1

In the first series of experiments we generate AR models with sparse inverse spectra by setting $B_0 = I$ and randomly choosing sparse lower triangular matrices B_k with entries ± 0.5. The random trials are continued until a stable AR model is found. The AR process is then used to generate N samples of the time series. The model dimensions are $n = 20$ and $p = 2$.

4.2.1 TOPOLOGY SELECTION

We first illustrate the basic topology selection method outlined above using the correct model order ($p = 2$). The sample size is $N = 512$.

Figure 2 shows the trade-off curve between the penalty $h_\infty(\mathrm{D}(X))$ and the log-likelihood $\mathcal{L}(X)$. We calculate the inverse spectra (10) for the computed points on the trade-off curve, and apply a threshold to them (as explained above, by setting entries with $\rho_{ij} \leq 10^{-1}$ to zero). The resulting topologies are shown in Figure 3. The patterns range from quite dense (small γ) to very sparse (large γ). The sparsity of the densest solution ($\gamma = 10^{-5}$) is identical to the sparsity of the least-squares estimate (i.e., the solution of the equations (15) with C given in (13) or, equivalently, the ML solution of (12) without the sparsity constraints). For each of the nine sparsity patterns, we solve the ML problem subject to sparsity constraints (16). We rank the nine solutions using the $\mathrm{AIC_c}$ and BIC scores defined in (20). Figure 4 shows the two scores and the negative log-likelihood as functions of γ. The models that minimize the $\mathrm{AIC_c}$/BIC scores turn out to be the same in this example (the models for $\gamma = 0.15$) and the corresponding topology is shown in Figure 5 (left). Only seven entries are misclassified (six entries are misclassified as zeros; one as nonzero). The sparsity pattern in the middle is the topology estimated by thresholding the partial coherence spectrum of

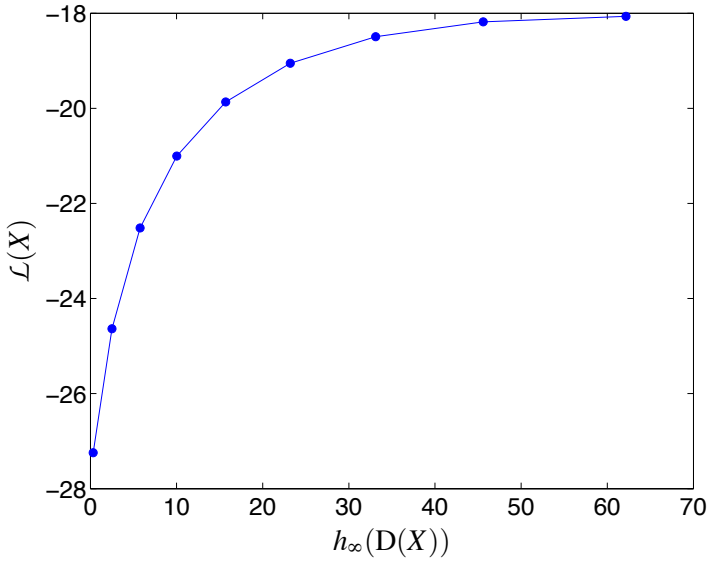


Figure 2: Trade-off curve between the log-likelihood $\mathcal{L}(X)$ and $h_\infty(\mathrm{D}(X))$.

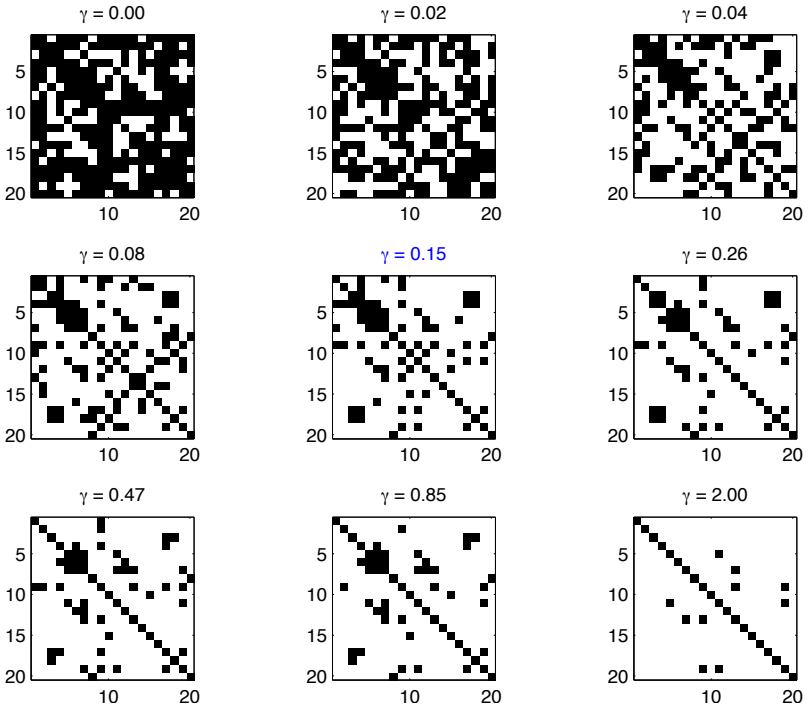


Figure 3: Topologies of solutions along the tradeoff curve in Figure 2 (ordered from right to left on the tradeoff curve).

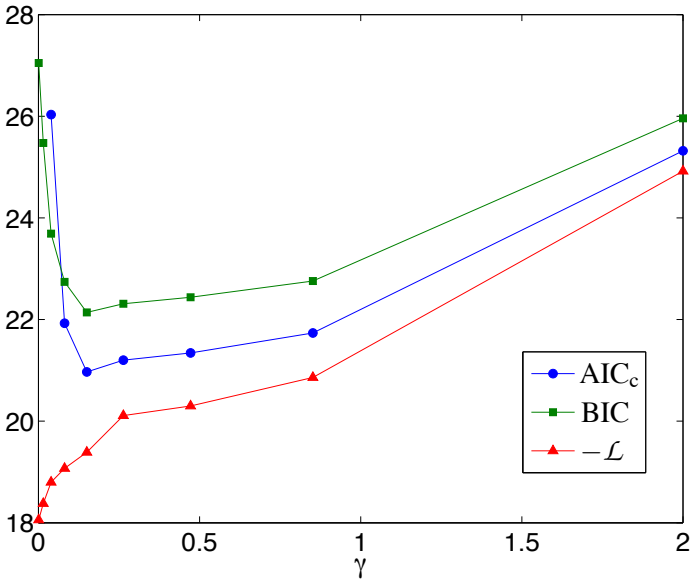


Figure 4: AIC$_c$ and BIC scores, and maximized log-likelihood for solutions on the trade-off curve in Figure 2.

the least-squares solution with the correct model order ($p = 2$). This pattern is computed by solving the ML problem (12) without constraints, and then thresholding the partial coherence (using the same threshold value 0.1 as in the other experiments). The difference between the two patterns clearly shows the benefits of the nonsmooth regularization for estimating a sparse topology. The sparsity pattern on the right of Figure 5 is obtained from the covariance selection method with ℓ_1-norm regularization (i.e., by setting $p = 0$ in the regularized ML problem (21)) and thresholding the partial coherence. Ignoring the model dynamics substantially increased the error in the topology selection.

4.2.2 COMPARISON WITH OTHER TYPES OF REGULARIZATION

To compare the quality of the sparse models with the models obtained from other estimation methods we evaluate the Kullback-Leibler (KL) divergence (Bach and Jordan, 2004) between the true and the estimated spectra as a function of the sample size N for the following six methods.

1. ML estimation without conditional independence constraints (or least-squares estimate). This is the solution of (12) without the constraints, and it can be computed by solving the normal equations (15).

2. ML estimation with conditional independence constraints determined by thresholding the partial coherence matrix of the least-squares estimate (solution 1).

3. ML estimation with Tikhonov regularization and without conditional independence constraints. Tikhonov regularization (also known as *ridge regression* or ℓ_2-regularization) is widely used in statistics and estimation (Hastie et al., 2009, §3.4). A Tikhonov-regularized ML estimate

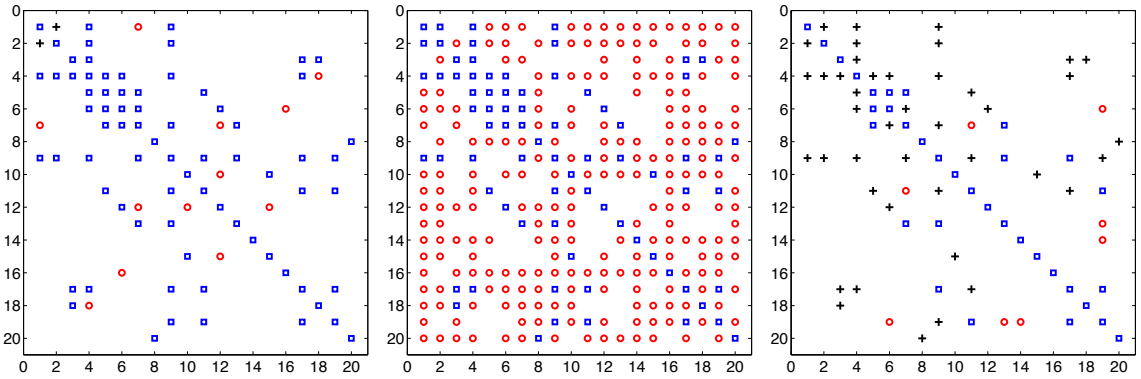


Figure 5: *Left*. The sparsity pattern from the regularized ML problem with $\gamma = 0.15$. *Middle*. The sparsity pattern estimated from the least-squares solution. *Right*. The sparsity pattern from the regularized ML problem for a static model ($p = 0$). The blue squares are the correctly identified nonzero entries (true positives). The red circles are the entries that are misclassified as nonzero (false positives). The black crosses are entries that are misclassified as zeros (false negatives).

is the solution of

$$\text{minimize} \quad -2\log\det B_0 + \mathbf{tr}(CB^TB) + \gamma\|B\|_F^2.$$

The solution can be computed from the normal equations (15) with C replaced by $C+\gamma I$. The solution of this problem can therefore also be viewed as a ML estimate using a perturbed sample covariance matrix $C+\gamma I$. In the experiment, the value of γ is determined by performing a five-fold cross-validation (Hastie et al., 2009, §7.10).

4. ML estimation with conditional independence constraints determined by thresholding the inverse spectral density for the Tikhonov estimate (solution 3).

5. Regularized ML estimation with h_∞-penalty. This is the solution of problem (21) with penalty function (22).

6. ML estimation with conditional independence constraints determined by thresholding the inverse spectral density for the h_∞-regularized ML estimate (solution 5).

The total number of variables in this example is $n(n+1)/2 + pn^2 = 1010$ variables. We show the results in Figure 6 in two different settings: with small sample sizes ($N < 1010$) and with moderate to large sample sizes ($N \geq 1010$). We can note that for small sample sizes N the constrained ML estimates (models 2,4,6) are not better than the unconstrained estimates (models 1,3,5), and much worse in the case of the Tikhonov-regularized estimates. This can be explained by large errors in the estimated topology. For larger N the constrained estimates are consistently better than the unconstrained models, and for very large N the three constrained ML estimates give the same accuracy. For small and moderate N we also see that model 6 (ML estimate for the topology selected via nonsmooth regularization) is much more accurate than the other methods.

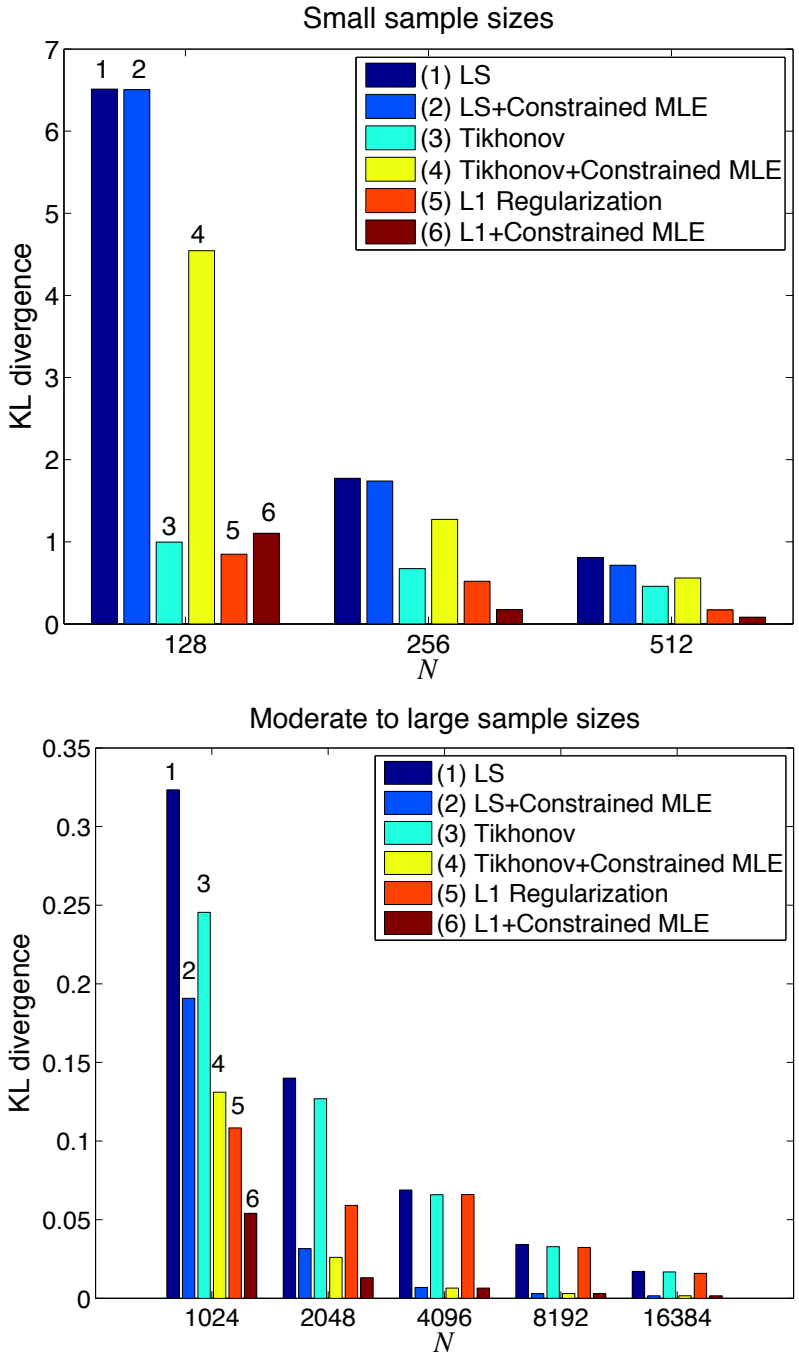


Figure 6: KL divergence between estimated AR models and the true model $(n = 20, p = 2)$ versus the number of samples N. We compare six methods: (1) least-squares estimate, (2) constrained ML estimate with topology estimated by thresholding solution 1, (3) ML estimate with Tikhonov regularization, (4) constrained ML estimate with topology estimated by thresholding solution 3, (5) regularized ML estimate with h_∞-penalty, (6) constrained ML estimate with topology estimated by thresholding solution 5.

4.2.3 ERRORS IN TOPOLOGY AS A FUNCTION OF SAMPLE SIZE

In the last figure (Figure 7) we examine how fast the error in the topology selection decreases with increasing sample length N for three topology selection methods: LS estimation followed by thresholding, ML estimation with Tikhonov regularization followed by thresholding, and ML estimation with nonsmooth regularization followed by thresholding. For each sample size N we show the errors averaged over 50 sample sequences (i.e., 50 different sample covariance matrices C). "False positives" refers to entries that are incorrectly classified as nonzeros (i.e., incorrectly added edges in the graphical model). "False negatives" are entries that are incorrectly classified as zeros (i.e., incorrectly deleted edges). The top graphs in Figure 7 show the fraction of false positives and false negatives versus the sample size. The bottom graphs show the total fraction of misclassified entries. We compare the three methods listed above. As can be seen, the total error in the estimated topology is reduced in the regularized estimates, and the errors decrease more rapidly when we regularize with the sum-of-norms penalty h_∞.

4.3 Experiment 2

In the second experiment we compare different penalty functions h for the regularized ML problem (21): the 'sum-of-ℓ_∞-norms' penalty h_∞ defined in (22), the 'sum-of-ℓ_2-norms' penalty h_2 defined in (28) with $\alpha = 2$, and the 'sum-of-ℓ_1-norms' penalty h_1 defined in (28) with $\alpha = 1$. These penalty functions all yield models with a sparse inverse spectrum

$$S(\omega)^{-1} = Y_0 + \frac{1}{2}\sum_{k=1}^{p}(e^{-\mathrm{j}k\omega}Y_k + e^{\mathrm{j}k\omega}Y_k^T),$$

but have different degrees of sparsity for the entries $(Y_k)_{ij}$ within each group i, j.

The data are generated by randomly choosing sparse coefficients Y_k of an inverse spectrum (10). For each (i, j) of nonzero locations in $S(\omega)^{-1}$, we select random values $(Y_k)_{ij}$ with about the same magnitude for all k. If necessary, a multiple of the identity matrix is added to Y_0 to guarantee the positiveness of the spectrum. An AR realization of the spectrum is then computed by spectral factorization and used to generate sample time series. The model dimensions are $n = 5$, $p = 7$.

Figure 8 shows typical values for the estimated coefficients $(Y_k)_{ij}$. The three penalty functions all give the same topology, but a different sparsity with the same group i, j of coefficients. The sparsity within each group is largest for the h_1-penalty and smallest for the h_∞-penalty.

Table 1 shows the results of topology selection with the three penalties, for sample size $N = 512$ and averaged over 50 sample sequences. The h_∞-penalty gives the models with the smallest KL divergence and smallest error in topology. This is to be expected, given the distribution of the nonzero coefficients $(Y_k)_{ij}$ in the AR models that were used to generate the data. The results also agree with a comparison of different norms in a composite penalty function (Zhao et al., 2009). In general the best choice of norm will depend on how the coefficients are distributed within each group.

5. Applications

This section presents two examples of real data sets to demonstrate how topology selection can facilitate studies of relationship in multivariate time series.

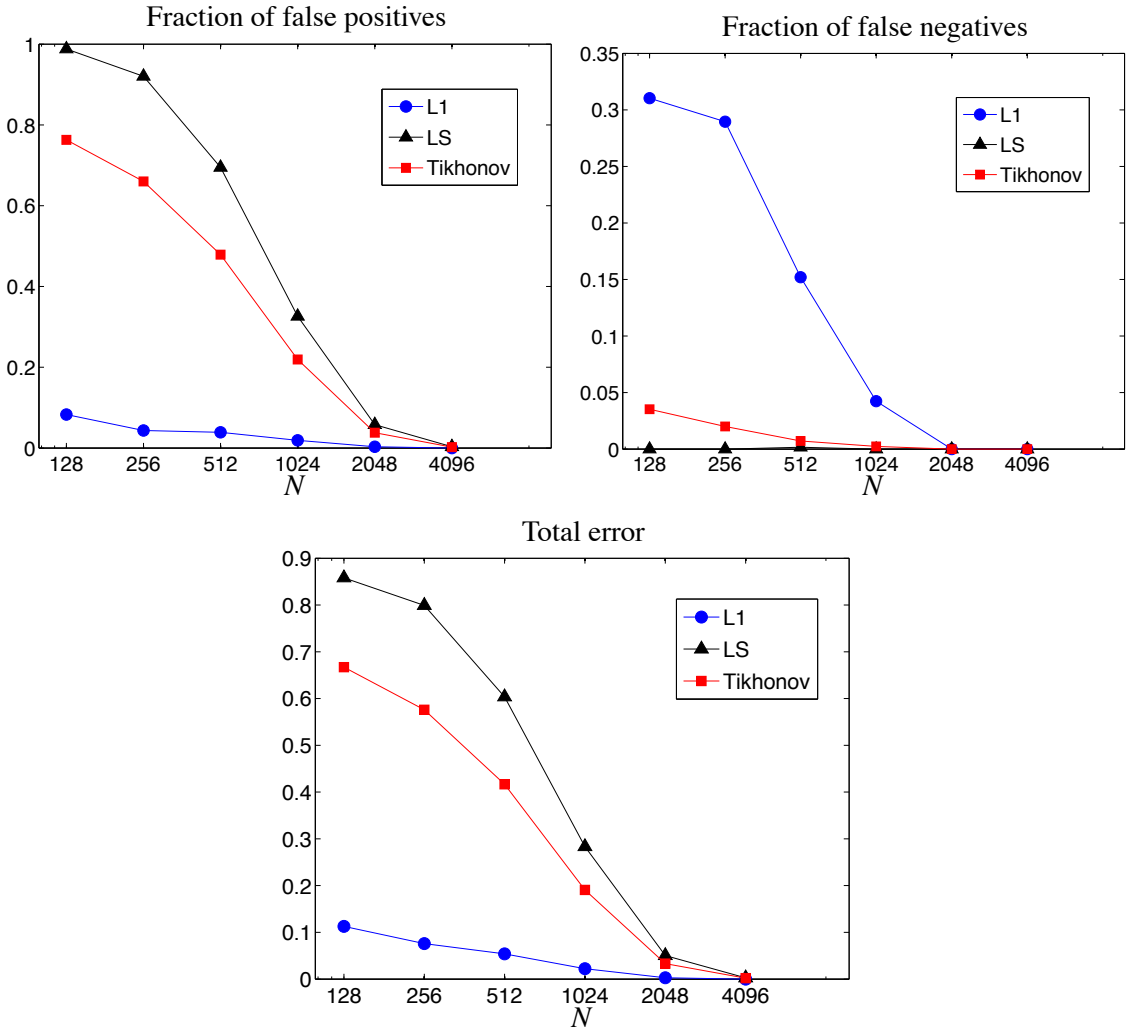


Figure 7: *Top left*. Fraction of incorrectly added edges in the estimated graph (number of upper triangular nonzeros in the estimated pattern that are incorrect, divided by the number of upper triangular zeros in the correct pattern). *Top right*. Fraction of incorrectly removed edges in the estimated graph (number of upper triangular zeros in the estimated pattern that are incorrect, divided by the number of upper triangular nonzeros in the correct pattern). *Bottom*. The combined classification error computed as the sum of the false positives and false negatives divided by the number of upper triangular entries in the pattern.

5.1 Functional Magnetic Resonance Imaging (fMRI) Data

In this section we apply the topology selection method to a functional magnetic resonance imaging (fMRI) time series. We use the StarPlus fMRI data set[1] (Mitchell et al., 2004), which was analyzed

1. StarPlus data can be found at `www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-81/www/`.

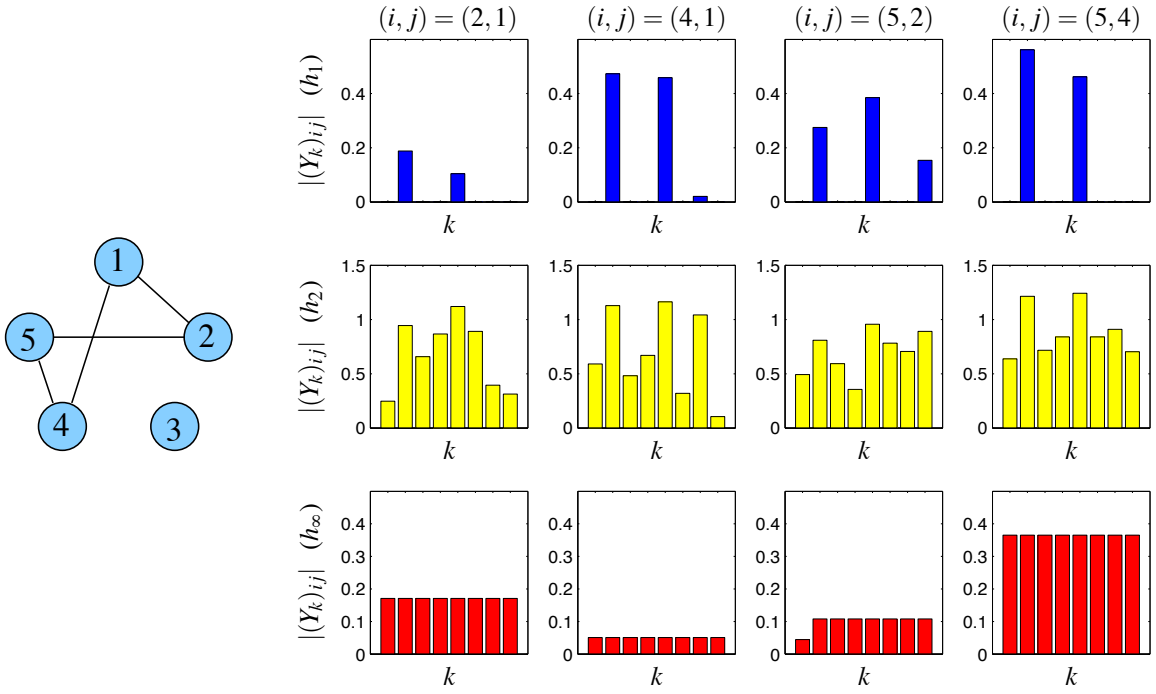


Figure 8: Nonzero coefficients $|(Y_k)_{ij}|$ for regularized ML estimates with penalty h_α, for $\alpha = 1, 2, \infty$.

Dimensions	KL divergence			Error in topology (%)		
	h_1	h_2	h_∞	h_1	h_2	h_∞
$n = 20, p = 2$	0.24	0.22	**0.21**	11.8	11.9	11.6
$n = 20, p = 4$	0.33	0.24	**0.19**	1.65	1.19	0.51
$n = 30, p = 2$	0.40	0.35	**0.30**	9.95	8.83	7.96
$n = 30, p = 4$	0.59	0.46	**0.40**	5.18	3.97	3.53

Table 1: Accuracy of topology selection methods with penalty h_α for $\alpha = 1, 2, \infty$. The table shows the average KL divergence with respect to the true model and the average percentage error in the estimated topology (defined as the sum of the false positives and false negatives divided by the number of upper triangular entries in the pattern), averaged over 50 instances.

using covariance selection in Scheinberg and Rish (2009). The data consists of 80 time series (runs) of brain image scans. In half of the 80 runs the input stimulus shown to the subject is a picture; in the other half it is a sentence. Each run contains 16 images, resulting in 640 images for each input. Mitchell et al. (2004) suggest a region of interest (ROI) of 1718 voxels. To reduce the dimension we took averages over groups of voxels in the ROI and considered four reduced graphs with $n = 7$, 50, 100, and 190 nodes, respectively.

We fit two different AR models, one for each input. The AR model orders selected by the BIC are shown in Table 2. As the problem size (n) becomes larger, the BIC tends to pick a static model

Input	$n=7$	$n=50$	$n=100$	$n=190$
Picture	$p=1$	$p=1$	$p=0$	$p=0$
Sentence	$p=1$	$p=1$	$p=0$	$p=0$

Table 2: AR model orders for the fMRI data set.

Input	Static models ($p=0$)			Time series models ($p=1$)		
	ℓ_1	Tikhonov	LS	ℓ_1	Tikhonov	LS
Picture	991	4116	4203	**0**	13467	13465
Sentence	922	4021	4131	**0**	13240	13238

Table 3: Relative BIC scores of six models fitted to two fMRI time series of size $n=50$. The 'static' models are Gaussian graphical models (i.e., AR models of order $p=0$), the time series models are AR models of order $p=1$. The models are constrained ML estimates with topologies estimated using three different methods: Regularized ML estimate with h_α-penalty, Tikhonov-regularized ML estimate, and the least-squares estimate. The BIC scores are relative to the score of the best model (time series models of regularized ML estimate with h_α-penalty).

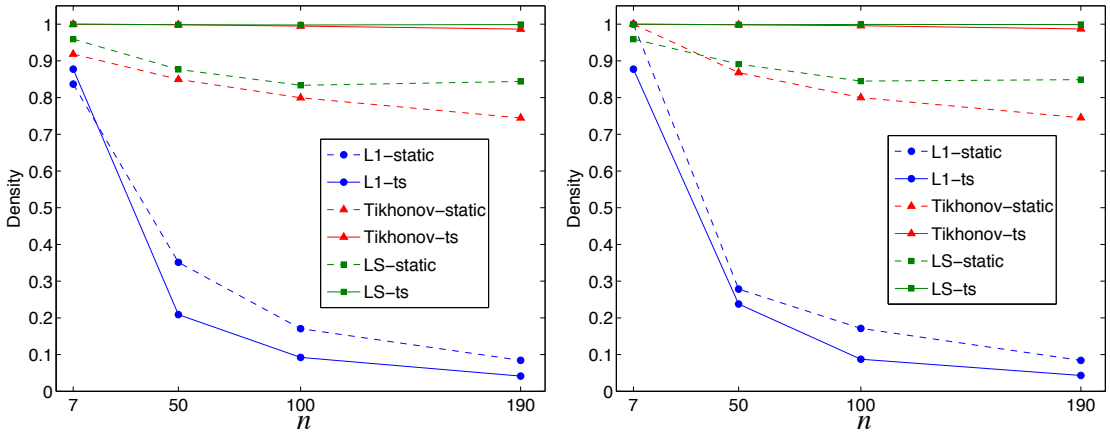


Figure 9: Density of the graphical models of fMRI data for 'picture' stimulus (*left*) and for 'sentence' stimulus (*right*). The density is computed as the number of nonzero entries in the estimated inverse spectrum divided by n^2.

($p=0$). Table 3 shows the BIC scores of different models for the experiment with size $n=50$.

The topologies selected by the BIC are the regularized ML estimates with h_∞-penalty. Figure 9 shows the sparsity of the estimated graphs from the least-squares, Tikhonov-regularized ML, and h_∞-regularized ML methods. The plots show that the h_∞-regularization produces much sparser graphs than the other two methods.

To get an idea of the accuracy of the estimated network structure, we validated the result with a simple classification experiment. For each input we keep one fMRI run as a test problem and use

model order	$n = 7$	$n = 50$	$n = 100$	$n = 190$
$p = 0$	0.21	0.16	**0.11**	**0.06**
$p = 1$	**0.20**	**0.16**	0.16	0.11

Table 4: Classification error of fMRI data versus model size. The error is the number of runs for which the stimulus input is correctly identified divided by the total number of runs (40).

the 39 remaining runs to estimate a sparse AR model. The two models are then used to guess the inputs shown to the subject during the test run. The classification algorithm computes the likelihood of each input, based on the two models, and selects the input with the highest likelihood. We repeat this for each of the 40 choices of test run. Table 4 shows the classification error versus the number of nodes in the graph. We see that the classification is quite successful and achieves an error in the range 6–20%. The error tends to be smaller if we use less averaging (larger n). We also note that for each n, the AR model of order p chosen in Table 2 also performs slightly better in the classification experiment.

5.2 International Stock Market Data

We consider a multivariate time series of 17 stock market indices: the S&P 5000 composite index (U.S.), Toronto stock exchange 300 index (Canada), the All ordinary composite stock index (Australia), the Nikkei 225 stock index (Japan), the Hang Seng stock composite index (Hong Kong), the FTSE 100 share index (United Kingdom), the Frankfurt DAX 30 composite index (German), the CAC 40 stock composite index (France), MIBTEL index (Italy), the Zurich Swiss Market composite index (Switzerland), the Amsterdam exchange index (Netherlands), the Austrian traded index (Austria), IBEX 35 (Spain), BEL 20 (Belgium), the OMX Helsinki 25 index (Finland), the Portugese stock index (Portugal), the Irish stock exchange index (Ireland). The data were stock index closing prices recorded from June 3, 1997 to June 30, 1999 and obtained from `www.globalfinancialdata.com`. The data were converted to US dollars. Missing data due to national holidays were replaced by the most recent values. For each market we use as variable the return between trading day $k-1$ and k, defined as

$$r_k = 100 \log(\pi_k / \pi_{k-1}),$$

where π_k is the closing price on day k. This results in 17-dimensional time series of length 540. Similar time series for a smaller number of markets were analyzed in Bessler and Yang (2003) and Abdelwahab et al. (2008).

We solve the h_∞-regularized ML problem with model orders ranging from $p = 0$ to $p = 3$, and for each value collect the topologies along the trade-off curve, as in the previous examples. The AIC_c and BIC criteria were then used to select a model. Both criteria selected a model of order $p = 1$ and the same sparsity pattern (corresponding to a value $\gamma = 0.34$). Figure 10 (right) shows ρ_{ij}, the maximum magnitude of the partial coherence of the model, and compares it with a thresholded nonparametric estimate obtained with Welch's method (Proakis, 2001) and the constrained ML model with topology obtained by thresholding the least-squares estimate. We note that the graph topologies suggested by the nonparametric and least-squares estimates are much denser than the regularized ML estimate.

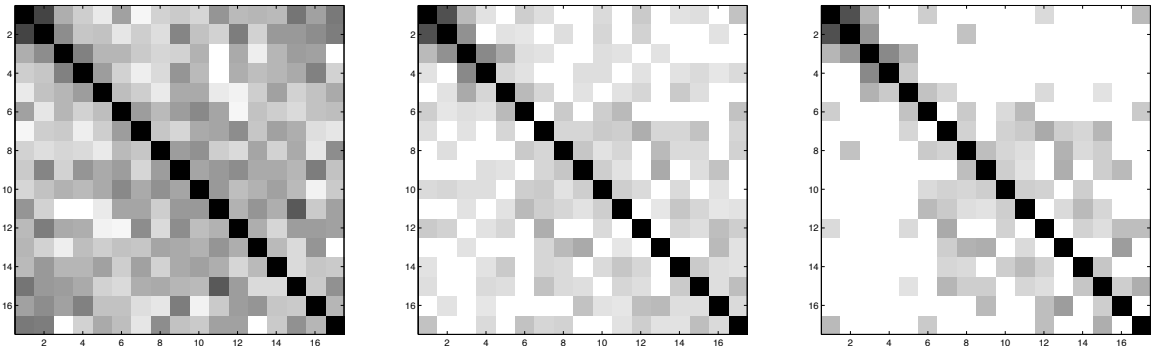


Figure 10: The maximum magnitude ρ_{ij} of the partial coherence for three models of the stock exchange data. *Left:* Thresholded nonparametric sample estimate using Welch's method. *Middle:* Constrained ML estimate with topology determined from the LS solution. *Right:* Constrained ML estimate with topology determined from the h_∞-regularized ML estimate.

Figure 11 shows the graphical model estimated by the h_∞-regularized ML problem. The thickness of the edges is proportional to ρ_{ij}. We recognize many connections that can be explained from geographic proximity or economic ties between the countries. For example, we see strong connections between the U.S. and Canada, between Australia, Japan, and Hong Kong, between Hong Kong and U.K., between the southern European countries, et cetera. Overall the graphical model seems plausible, and the experiment suggests that the topology selection method is quite effective.

6. First-order Optimization Algorithms

In the preceding sections we have considered four convex optimization problems. The constrained ML estimation problem (16) and its dual (17) have differentiable objectives and linear equality and matrix inequality constraints. The regularized ML problem (21) also includes a nondifferentiable term in the objective, and its dual (27) has a differentiable objective but constraints that involve nondifferentiable functions. These optimization problems can be solved by interior-point methods, for example, the path-following methods developed for convex determinant maximization problems (Toh, 1999; Vandenberghe et al., 1998). In practice, however, the problems are often too large for interior-point methods because they involve matrix variables (X or Z) of high dimension. In this section we therefore investigate less expensive first-order algorithms applied to a reformulation of the dual problems (17) and (27). The dual approach avoids several difficulties that arise in first-order methods applied to the primal problems: the complicated constraints in the constrained ML problem (16), the fact that its objective, which is also the first term in the objective of the regularized ML problem (21), is not strictly convex, the nondifferentiability of the penalty term in (21), and, most important, the fact the solution X has low rank and therefore lies on the boundary of the feasible set. (For the regularized ML problem (21), these difficulties could be addressed as in the covariance selection method of Banerjee et al. (2008), by applying Nesterov's fast gradient method to an approximation of the primal problem with a smoothed objective and a closed bounded constraint set (Nesterov, 2005). In our limited experience, with a fixed and conservative choice of

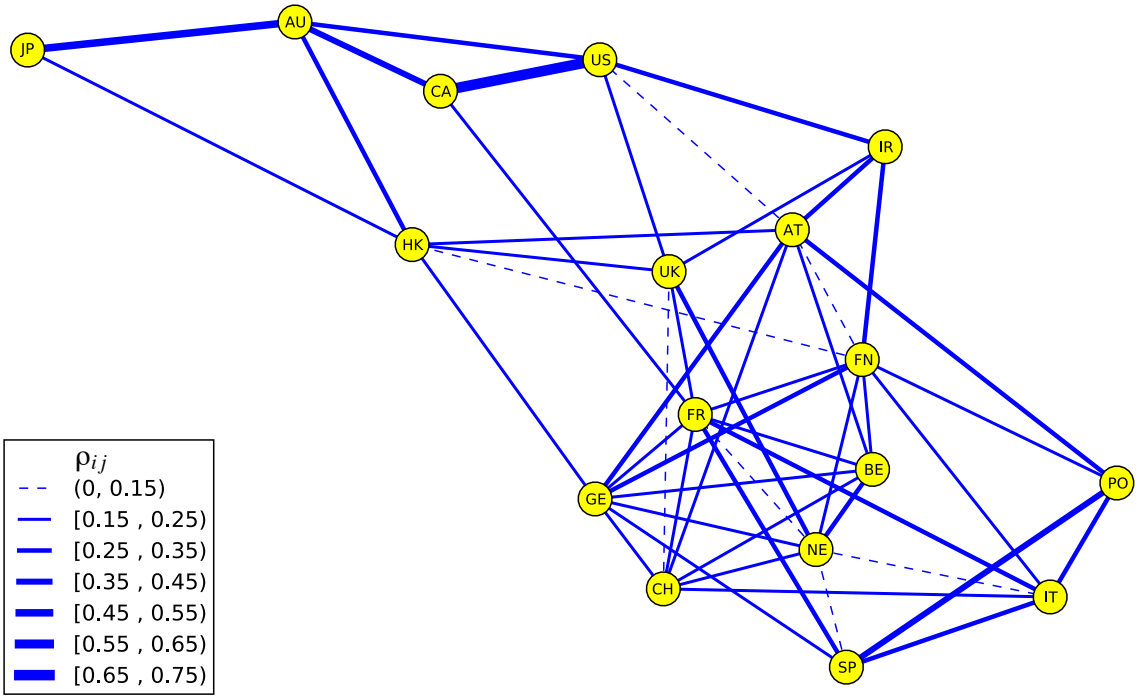


Figure 11: A graphical model of stock market data. The strength of connections is represented by the width of the blue links, which is proportional to $\rho_{ij} = \sup_\omega |R(\omega)_{ij}|$ if it is greater than 0.15.

the smoothing and bounding parameters, this algorithm was slower than the dual gradient projection method described in this section, so we will not pursue it here.)

6.1 Reformulated Dual Problems

To reformulate the dual problems we eliminate the variable W in (17) and (27). Let $V = C + \mathrm{T}(\mathrm{P}(Z))$, respectively, $V = C + \mathrm{T}(Z)$. The inequality

$$V - \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} V_{00} - W & V_{1:p,0}^T \\ V_{1:p,0} & V_{1:p,1:p} \end{bmatrix} \succeq 0,$$

is equivalent to

$$V_{1:p,1:p} \succeq 0, \qquad \mathrm{range}(V_{1:p,0}) \subseteq \mathrm{range}(V_{1:p,1:p}), \qquad V_{00} - V_{1:p,0}^T V_{1:p,1:p}^\dagger V_{1:p,0} \succeq W, \tag{32}$$

where $V_{1:p,1:p}^\dagger$ is the pseudo-inverse of $V_{1:p,1:p}$. If $V \succeq 0$, then the matrix W with maximum determinant that satisfies (32) is equal to $V_{00} - V_{1:p,0}^T V_{1:p,1:p}^\dagger V_{1:p,0}$, the *Schur complement* of $V_{1:p,1:p}$ in V. This observation allows us to eliminate W from (17) and (27). Problem (17) can be written as an unconstrained problem

$$\text{maximize} \quad -\phi(C + \mathrm{T}(\mathrm{P}(Z))), \tag{33}$$

and problem (27) as a problem with simple constraints

$$\begin{array}{ll} \text{maximize} & -\phi(C+\mathrm{T}(Z)) \\ \text{subject to} & \sum_{k=0}^{p}(|(Z_k)_{ij}|+|(Z_k)_{ji}|) \leq \gamma, \quad i \neq j \\ & \mathbf{diag}(Z_k) = 0, \quad k = 0,\ldots,p. \end{array} \tag{34}$$

Here $\phi : \mathbf{S}^{n(p+1)} \to \mathbf{R}$ is defined as

$$\phi(V) = -\log\det\left(V_{00} - V_{1:p,0}^T V_{1:p,1:p}^\dagger V_{1:p,0}\right) - n,$$

with domain $\mathbf{dom}\,\phi = \{V \in \mathbf{S}_+^{n(p+1)} \mid V_{00} - V_{1:p,0}^T V_{1:p,1:p}^\dagger V_{1:p,0} \succ 0\}$. This function is convex, since it can be expressed as

$$\phi(V) = \inf\left\{ -\log\det W \;\middle|\; \left[\begin{array}{cc} W & 0 \\ 0 & 0 \end{array}\right] \preceq V \right\} - n,$$

and convexity of this expression follows from results in convex analysis (Boyd and Vandenberghe, 2004, §3.2.5). It is also a smooth function on the interior of its domain and its gradient at a positive definite V can be expressed as

$$\nabla\phi(V) = -V^{-1} + \left[\begin{array}{cc} 0 & 0 \\ 0 & V_{1:p,1:p}^{-1} \end{array}\right]. \tag{35}$$

This can be seen, for example, from the identity $\det V = \det V_{1:p,1:p} \det(V_{00} - V_{1:p,0}^T V_{1:p,1:p}^{-1} V_{1:p,0})$, which gives $\phi(V) = -\log\det V + \log\det V_{1:p,1:p} - n$, and the fact that the gradient of $\log\det X$ is X^{-1}.

If $V = C + \mathrm{T}(\mathrm{P}(Z)) \succ 0$ at the optimum of (33) then the primal optimal solution can be computed from Z via the expressions

$$X = V^{-1} - \left[\begin{array}{cc} 0 & 0 \\ 0 & V_{1:p,1:p}^{-1} \end{array}\right] = \left[\begin{array}{c} -I \\ V_{1:p,1:p}^{-1} V_{1:p,0} \end{array}\right] W^{-1} \left[\begin{array}{c} -I \\ V_{1:p,1:p}^{-1} V_{1:p,0} \end{array}\right]^T \tag{36}$$

where $V = C + \mathrm{T}(\mathrm{P}(Z))$ and $W = V_{00} - V_{1:p,0}^T V_{1:p,1:p}^{-1} V_{1:p,0}$. The expression for X follows from the optimality condition (18) and the identities

$$V = \left[\begin{array}{cc} V_{00} - V_{1:p,0}^T V_{1:p,1:p}^{-1} V_{1:p,0} & 0 \\ 0 & 0 \end{array}\right] + \left[\begin{array}{c} V_{1:p,0}^T V_{1:p,1:p}^{-1} \\ I \end{array}\right] V_{1:p,1:p} \left[\begin{array}{c} V_{1:p,0}^T V_{1:p,1:p}^{-1} \\ I \end{array}\right]^T,$$

$$V^{-1} = \left[\begin{array}{cc} 0 & 0 \\ 0 & V_{1:p,1:p}^{-1} \end{array}\right] + \left[\begin{array}{c} -I \\ V_{1:p,1:p}^{-1} V_{1:p,0} \end{array}\right] (V_{00} - V_{1:p,0}^T V_{1:p,1:p}^{-1} V_{1:p,0})^{-1} \left[\begin{array}{c} -I \\ V_{1:p,1:p}^{-1} V_{1:p,0} \end{array}\right]^T. \tag{37}$$

The formula for V^{-1} also provides an alternative form of the gradient (35).

Similarly, if $C + \mathrm{T}(Z) \succ 0$ at the optimum of (34) then the primal optimal X can be computed from (36) with $V = C + \mathrm{T}(Z)$.

The reformulated dual problems are interesting because they can often be solved by gradient algorithms for unconstrained optimization or gradient projection algorithms for problems with simple constraints. To explain this, we again distinguish between Toeplitz and non-Toeplitz C. If C is

block-Toeplitz, then it can be shown that the functions $\phi(C+\mathrm{T}(\mathrm{P}(Z)))$ and $\phi(C+\mathrm{T}(Z))$ are *closed* convex functions (i.e., with closed sublevel sets) and that their domains are open. Consider the function ϕ restricted to the set of block-Toeplitz matrices, that is, $\phi(\mathrm{T}(R))$, where $R \in \mathbf{M}^{n,p}$. By definition, R is in the domain of $\phi(\mathrm{T}(R))$ if $\mathrm{T}(R) \succeq 0$ and there exists a positive definite W with

$$\mathrm{T}(R) \succeq \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix}.$$

From the property of block-Toeplitz matrices mentioned in Section 2.3, this implies $\mathrm{T}(R) \succ 0$. In other words, the domain of $\phi(\mathrm{T}(R))$ is the open set $\{R \mid \mathrm{T}(R) \succ 0\}$. By a similar argument, if a sequence of matrices R in the domain of $\phi(\mathrm{T}(R))$ converges to a point $\bar{R}$ in the boundary of the domain, then the Schur complement of $\mathrm{T}(\bar{R})_{1:p,1:p}$ in $\mathrm{T}(\bar{R})$ must be singular, and hence $\phi(\mathrm{T}(R)) \to \infty$. For a continuous function with an open domain this is equivalent to closedness (Boyd and Vandenberghe, 2004, p.639).

If C is not block-Toeplitz, then the functions $\phi(C+\mathrm{T}(\mathrm{P}(Z)))$ and $\phi(C+\mathrm{T}(Z))$ are not necessarily closed, and their domains not necessarily open. One implication is that it is possible that the optimal solution of (33) or (34) is at a point in the boundary of the domain of the cost function, that is, a point where $C+\mathrm{T}(\mathrm{P}(Z))$ or $C+\mathrm{T}(Z)$ are singular. However in practice, C is usually approximately block-Toeplitz and one can expect that the functions are often closed. Moreover, in order to apply unconstrained minimization algorithms it is sufficient that the algorithm is started at a point $Z^{(0)}$ for which the sublevel set $\{Z \mid \phi(C+\mathrm{T}(\mathrm{P}(Z))) \leq \phi(C+\mathrm{T}(\mathrm{P}(Z^{(0)})))\}$ is closed. This condition is considerably weaker than the requirement that all sublevel sets are closed.

6.2 Gradient Projection Algorithms

We now present some details on first-order algorithms for the reformulated dual problems. We focus on the constrained problem (34) since the unconstrained problem (33) can be handled as a special case. We first describe a version of the classical gradient projection with a backtracking line search (Polyak, 1987; Bertsekas, 1999). To simplify the notation we will use a generic problem format

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is convex and continuously differentiable with an open domain, and $\mathcal{C}$ is a closed convex set. We assume that a feasible point $x^{(0)}$ is known and that the sublevel set

$$\mathcal{S} = \{x \in \mathbf{dom}\, f \cap \mathcal{C} \mid f(x) \leq f(x^{(0)})\}$$

is closed and bounded. The closedness assumption is satisfied if f is a closed function. (See the previous paragraph on the validity of this assumption for problems (33) and (34).) We assume that projections on $\mathcal{C}$ are inexpensive and we denote the projection operator by $\mathcal{P}$:

$$\mathcal{P}(y) = \operatorname*{argmin}_{x \in \mathcal{C}} \|x-y\|_2.$$

The *gradient map* associated with f and $\mathcal{C}$ is defined as

$$G_t(x) = \frac{1}{t}\left(x - \mathcal{P}(x - t\nabla f(x))\right)$$

for $t > 0$. For an unconstrained problem, the gradient map is $G_t(x) = \nabla f(x)$, independent of t.

6.2.1 BASIC GRADIENT PROJECTION

The basic gradient projection method starts at $x^{(0)}$ and continues the iteration

$$\begin{aligned} x^{(k)} &= \mathcal{P}\left(x^{(k-1)} - t_k \nabla f(x^{(k-1)})\right) \\ &= x^{(k-1)} - t_k G_{t_k}(x^{(k-1)}) \end{aligned} \tag{38}$$

until a stopping criterion is satisfied. A classical convergence result states that $x^{(k)}$ converges to an optimal solution if t_k is fixed and equal to $1/L$, where L is a constant that satisfies

$$\|\nabla f(u) - \nabla f(v)\|_2 \leq L\|u - v\|_2 \quad \forall u, v \in \mathcal{S}, \tag{39}$$

(Polyak, 1987, §7.2.1). Although our assumptions ($\mathcal{S}$ is closed and bounded, and $\mathbf{dom}\, f$ is open) imply that the Lipschitz condition (39) holds for some constant $L > 0$, its value is not known in practice, so the fixed step size rule $t_k = 1/L$ cannot be used. We therefore determine t_k using a backtracking search (Beck and Teboulle, 2009). The step size search algorithm in iteration k starts at a value $t_k := \bar{t}_k$ where

$$\bar{t}_k = \min\left\{\frac{s^T s}{s^T y}, t_{\max}\right\}, \tag{40}$$

where

$$s = x^{(k-1)} - x^{(k-2)}, \qquad y = \nabla f(x^{(k-1)}) - \nabla f(x^{(k-2)}),$$

and $t_{\max}$ is a positive constant. (In the first iteration we initialize the step size as $t_1 = t_{\max}$.) The search then repeats the update $t_k := \beta t_k$ (where $\beta \in (0,1)$ is an algorithm parameter) until $x^{(k-1)} - t_k G_{t_k}(x^{(k-1)}) \in \mathbf{dom}\, f$ and

$$f(x^{(k-1)} - t_k G_{t_k}(x^{(k-1)})) \leq f(x^{(k-1)}) - t_k \nabla f(x^{(k-1)})^T G_{t_k}(x^{(k-1)}) + \frac{t_k}{2}\|G_{t_k}(x^{(k-1)})\|_2^2. \tag{41}$$

The resulting step size t_k is used in the update to $x^{(k)}$ in (38). Note that the trial points

$$x^{(k-1)} - t_k G_{t_k}(x^{(k-1)}) = \mathcal{P}\left(x^{(k-1)} - t_k \nabla f(x^{(k-1)})\right)$$

generated during the step size search are not necessarily on a straight line. The trajectory is sometimes referred to as the *projection arc* (Bertsekas, 1999, §2.3).

The step length $\|s\|_2^2/s^T y$ is known as the *Barzilai-Borwein* step size and forms the basis of *spectral gradient* methods (Barzilai and Borwein, 1988; Birgin et al., 2003; Figueiredo et al., 2007; Wright et al., 2009). It can be motivated by the easily established fact that $\|s\|_2^2/s^T y \geq 1/L$ if f satisfies (39), so it is a readily computed upper bound for $1/L$.

6.2.2 VARIATIONS

The basic gradient projection method can be varied in several ways, some of which will be compared in the numerical experiments below. To avoid computing a projection for each trial step size t_k in the step size search, we can replace the gradient update with

$$x^{(k)} = x^{(k-1)} - t_k G_{\bar{t}_k}(x^{(k-1)}) \tag{42}$$

where $\bar{t}_k$ is held fixed at the value (40) and t_k is determined by a backtracking search: we take $t_k := \bar{t}_k$ and then backtrack ($t_k := \beta t_k$) until $x^{(k-1)} - t_k G_{\bar{t}_k}(x^{(k-1)}) \in \mathbf{dom}\, f$ and

$$f(x^{(k-1)} - t_k G_{\bar{t}_k}(x^{(k-1)})) \leq f(x^{(k-1)}) - t_k \nabla f(x^{(k-1)})^T G_{\bar{t}_k}(x^{(k-1)}) + \frac{t_k}{2} \|G_{\bar{t}_k}(x^{(k-1)})\|_2^2. \tag{43}$$

In this method the trial points during the step size selection follow a straight line, and each step only requires a function evaluation.

Many alternatives to the step size rules (38) and (42) are available in the literature, for example, the Armijo rule (Bertsekas, 1999, §2.3), and conditions that allow non-monotone convergence (Birgin et al., 2000; Lu and Zhang, 2009). In our experiments these variations gave similar results as the step size rules outlined above.

Another attractive class of gradient projection algorithms are the optimal first-order methods originated by Nesterov (Nesterov, 2004; Tseng, 2008; Beck and Teboulle, 2009). For functions whose gradient is Lipschitz continuous on $\mathcal{C}$, these algorithms have a better complexity than the classical gradient projection method (at most $O(\sqrt{1/\varepsilon})$ iterations are needed to reach an accuracy ε, as opposed to $O(1/\varepsilon)$ for the gradient projection method). These theoretical complexity results are valid if a constant step size $t_k = 1/L$ is used where L is the Lipschitz constant for the gradient, or if the step sizes form an nonincreasing sequence ($t_{k+1} \leq t_k$) determined by a backtracking line search (Beck and Teboulle, 2009; Tseng, 2008). The assumption that the gradient is Lipschitz continuous on $\mathcal{C}$ does not hold for the problem considered here, and it is not clear if the convergence analysis can be extended to the case when the gradient is Lipschitz continuous only on the initial sublevel set. Nevertheless, an implementation with a backtracking line search worked well in our experiments (see next section).

6.2.3 IMPLEMENTATION DETAILS

The most important steps in the gradient projection algorithms applied to (33) are the evaluations of the gradient of the objective function and the projections on the set defined by the constraints. We now explain these two steps and the stopping criterion in more detail.

The gradient (35) of ϕ at a point V can be evaluated from a Cholesky factorization $V = L^T L$ with L lower triangular. If we partition L as

$$L = \begin{bmatrix} L_{00} & 0 \\ L_{1:p,0} & L_{1:p,1:p} \end{bmatrix}$$

then

$$\nabla\phi(V) = \begin{bmatrix} I \\ -L_{1:p,1:p}^{-1} L_{1:p,0} \end{bmatrix} L_{00}^{-1} L_{00}^{-T} \begin{bmatrix} I \\ -L_{1:p,1:p}^{-1} L_{1:p,0} \end{bmatrix}^T.$$

The projection $\mathcal{P}(U)$ of a matrix $U \in \mathbf{M}^{p,n}$ on the set defined by the constraints in (34) can be efficiently computed as follows. Clearly, the diagonal entries of $\mathcal{P}(U)_k$ are zero for $k = 0, \ldots, p$. To find the off-diagonal entries we can solve an independent problem

$$\begin{array}{ll} \text{minimize} & 2((Z_0)_{ij} - (U_0)_{ij})^2 + \sum_{k=1}^{p} \left(((Z_k)_{ij} - (U_k)_{ij})^2 + ((Z_k)_{ji} - (U_k)_{ji})^2\right) \\ \text{subject to} & \sum_{k=0}^{p} (|(Z_k)_{ij}| + |(Z_k)_{ji}|) \leq \gamma \end{array}$$

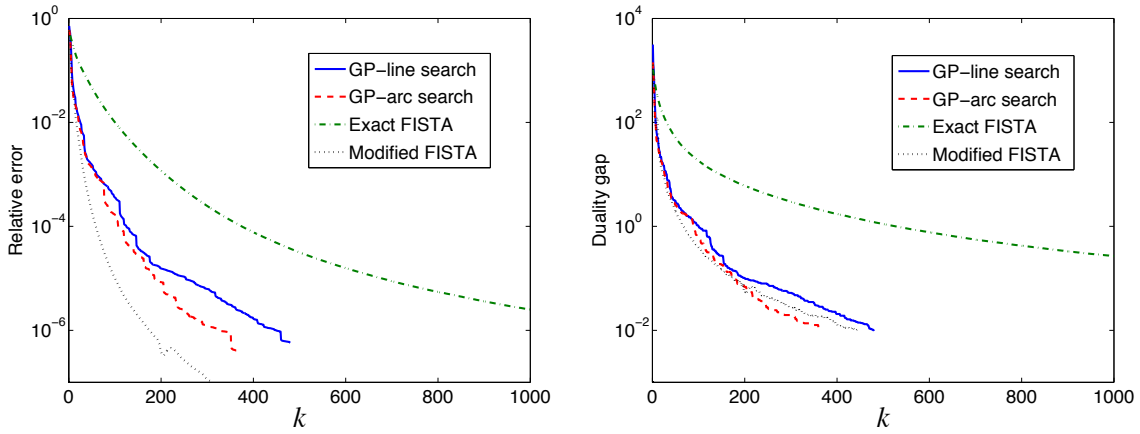


Figure 12: Convergence of gradient projection algorithms. *Left:* Relative error $(f(Z^{(k)})-f^\star)/|f^\star|$ versus the number of iterations. *Right:* Duality gap versus the number of iterations.

for each i,j with $j>i$. This is the problem of projecting a vector on the ℓ_1-norm ball. The solution is easily derived from duality and can be calculated by applying to the entries $(U_k)_{ij}$ the shrinkage operation familiar in sparse optimization (see, for example, Tibshirani, 1996).

The following stopping criterion will be used in the experiments. At each iteration, we compute X in (36) from the current iterate Z. This matrix X is primal feasible, as can be seen from the identity (37) and the fact that $C+\mathrm{T}(Z)\succ 0$. By taking the Schur complement of $(C+\mathrm{T}(Z))_{1:p,1:p}$ we also find a dual feasible W in (27). The duality gap between this primal feasible X and the dual feasible Z,W is

$$\begin{aligned}\eta &= -\log\det X_{00}+\mathbf{tr}(CX)+\gamma h(\mathrm{D}(X))-\log\det W-n\\ &= \mathbf{tr}(CX)-n+\gamma h(\mathrm{D}(X))\\ &= \mathbf{tr}((C+\mathrm{T}(Z))X)-n-\mathbf{tr}(X\,\mathrm{T}(Z))+\gamma h(\mathrm{D}(X))\\ &= -\mathbf{tr}(X\,\mathrm{T}(Z))+\gamma h(\mathrm{D}(X)).\end{aligned}$$

We terminate when the duality gap is below a given tolerance.

6.3 Numerical Example

We generate AR models as in the experiment described in Section 4.2. In the first experiment, the model dimensions are $n=300$, $p=2$, $N=2n(p+1)$. The true inverse spectrum has 10428 non-zero entries in the upper triangular part (a density of about 12%). The penalty parameter γ is set at $\gamma=0.1$. The variable Z in the reformulated dual problem (34) is a matrix in $\mathbf{M}^{300,2}$, so the problem has $n(n+1)/2+pn^2=225150$ optimization variables. We start the gradient projection algorithm at a strictly feasible $Z^{(0)}=0$, and terminate when the duality gap is below 10^{-2} (the optimal value is on the order of hundreds).

Figure 12 shows the relative error $(f(Z^{(k)})-f^\star)/|f^\star|$ where $f(Z)=\phi(C+\mathrm{T}(\mathrm{P}(Z)))$ and $f^\star$ is the optimal value. It also shows the duality gap $\eta^{(k)}$ versus the iteration number for a typical instance. 'GP with arc search' refers to the gradient projection method (38) with step size rule (41).

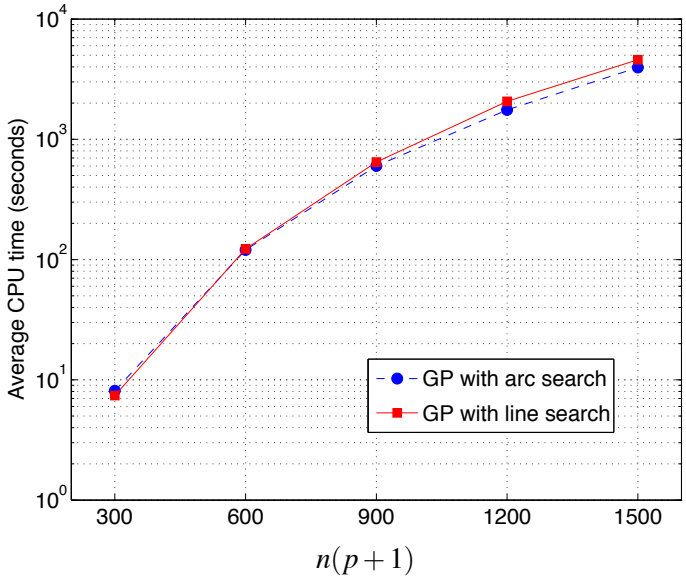


Figure 13: Average CPU times (averaged over 10 runs) of the gradient projection algorithm versus the problem size. The algorithm stops when the duality gap is less than 10^{-1}. The red squares correspond to 'GP with line search' and the blue squares correspond to 'GP with arc search'.

'GP with line search' refers to the gradient projection method (42) with step size rule (43). The step size searches required at most 15 backtracking steps to find an acceptable step size. As can be seen, a solution with a moderate accuracy (relative error in the range 10^{-4}–10^{-3}) is obtained after a number of iterations that is only a fraction of the problem size. The convergence of the 'arc search' method is slightly faster, but it should be kept in mind that this method is more expensive than the 'line search'.

The 'Exact FISTA' method is the gradient projection algorithm with backtracking line search from Beck and Teboulle (2009) using monotonically decreasing step sizes ($t_k \leq t_{k-1}$, as required by the theory in Beck and Teboulle 2009). As can be seen the convergence was not faster than the classical gradient projection method. A heuristic modification in which the step sizes are not forced to be nonincreasing, but at each iteration the line searche is initialized at the Barzilai and Borwein steplength (40), was often about five times faster. This algorithm is referred to as 'Modified FISTA' in the figure.

Figure 13 shows the CPU time versus problem size on a 3GHz Intel Pentium(R) 4 processor with 2.94 GB of RAM, for the 'GP with arc search' and 'GP with line search' algorithms. The test problems are generated as in the previous experiment, with $p = 2$ and varying n. The algorithms stop when it achieves a duality gap less than $\varepsilon = 0.1$. This yields a solution with a moderate accuracy (relative gap in the range 10^{-4}–10^{-3}). The plot shows that the times needed to solving the regularized ML estimation using both algorithms are fairly comparable with a slight advantage for 'GP with arc search' when n is large. Although the backtracking steps in the arc search method are more expensive, the gradient projection method with this step size selection required fewer iterations in most cases.

7. Conclusion

We have presented a convex optimization method for topology selection in graphical models of autoregressive Gaussian processes. The method is based on augmenting the maximum likelihood estimation problem with an ℓ_1-type penalty function, chosen to promote sparsity in the inverse spectrum. By tracing the trade-off curve between the log-likelihood and the penalty function, we obtain a small set of sparse graph topologies, that can then be ranked according to information-theoretic criteria such as the AIC or BIC. This procedure avoids the combinatorial complexity of enumerating all possible topologies, and produces accurate results for smaller sample sizes than methods based on empirical or least-squares estimates. To solve the large, nonsmooth convex optimization problems that result from this formulation, we have investigated a gradient projection method applied to a reformulated dual problem. Experiments with randomly generated examples, and an analysis of an fMRI time series and a time series of international stock market indices were included to confirm the effectiveness of this approach.

Acknowledgments

The authors thank Zhaosong Lu for interesting discussions on algorithms for the penalized ML problem. This research was supported by NSF under grant ECCS-0824003 and by a Royal Thai government scholarship.

References

A. Abdelwahab, O. Amor, and T. Abdelwahed. The analysis of the interdependence structure in international financial markets by graphical models. *International Research Journal of Finance and Economics*, 15:291–306, 2008.

N. Bani Asadi, I. Rish, K. Scheinberg, D. Kanevsky, and B. Ramabhadran. A MAP approach to learning sparse Gaussian markov networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1721–1724, 2009.

F. R. Bach and M. I. Jordan. Learning graphical models for stationary time series. *IEEE Transactions on Signal Processing*, 32:2189–2199, 2004.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Nashua, New Hampshire, second edition, 1999.

D.A. Bessler and J. Yang. The structure of interdependence in international stock markets. *Journal of International Money and Finance*, 22(2):261–287, 2003.

E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM J. on Optimization*, 10(4):1196–1211, 2000.

E. G. Birgin, J. M. Martínez, and M. Raydan. Inexact spectral projected gradient methods on convex sets. *IMA Journal of Numerical Analysis*, 23:539–559, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004. `www.stanford.edu/~boyd/cvxbook`.

D. R. Brillinger. *Time series: Data analysis and theory*. Holden-Day, San Francisco, California, expanded edition, 1981.

J. Dahl, V. Roychowdhury, and L. Vandenberghe. Maximum-likelihood estimation of multivariate normal graphical models: large-scale numerical implementation and topology selection. Technical report, Electrical Engineering Department, UCLA, 2005.

R. Dahlhaus. Graphical interaction models for multivariate time series. *Metrika*, 51(2):157–172, 2000.

R. Dahlhaus, M. Eichler, and J. Sandkühler. Identification of synaptic connections in neural ensembles by graphical models. *Journal of Neuroscience Methods*, 77(1):93–107, 1997.

A. P. Dempster. Covariance selection. *Biometrics*, 28:157–175, 1972.

J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse Gaussians. In *Proceeding of the Conference on Uncertainty in AI*, 2008.

M. Eichler. Fitting graphical interaction models to multivariate time serie. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

M. Eichler. Testing nonparametric and semiparametric hypotheses in vector stationary processes. *Journal of Multivariate Analysis*, 99(5):968–1009, 2008.

M. Eichler, R. Dahlhaus, and J. Sandkühler. Partial correlation analysis for the identification of synaptic connections. *Biological Cybernetics*, 89(4):289–302, 2003.

S. Feiler, K.G. Müller, A. Müller, R. Dahlhaus, and W. Eich. Using interaction graphs for analysing the therapy process. *Psychother Psychosom*, 74(2):93–99, 2005.

M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

R. Fried and V. Didelez. Decomposability and selection of graphical models for multivariate time series. *Biometrika*, 90(2):251–267, 2003.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432, 2008.

U. Gather, M. Imhoff, and R. Fried. Graphical models for multivariate time series from intensive care monitoring. *Statistics in Medicine*, 21(18):2685–2701, 2002.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, New York, 2nd edition, 2009.

J. Z. Huang, N. Liu, M. Pourahmadi, and L. Liu. Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93(1):85–98, 2006.

Y. Kim, J. Kim, and Y. Kim. Blockwise sparse regression. *Statistica Sinica*, 16(2):375–390, 2006.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, 1996.

K. Lounici. Sup-norm convergence rate and sign concentration property of Lasso and Dantzig estimators. *Electronic Journal of Statistics*, 2:90–102, 2008.

Z. Lu. Smooth optimization approach for sparse covariance selection. *SIAM Journal on Optimization*, 19:1807, 2009.

Z. Lu. Adaptive first-order methods for general sparse inverse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 2010. forthcoming.

Z. Lu and Y. Zhang. An augmented lagrangian approach for sparse principal component analysis. 2009. Submitted.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

T.M. Mitchell, R. Hutchinson, R.S. Niculescu, F. Pereira, and X. Wang. Learning to decode cognitive states from brain images. *Machine Learning*, 57(1):145–175, 2004.

Y. Nesterov. *Introductory Lectures on Convex Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming Series A*, 103:127–152, 2005.

B. T. Polyak. *Introduction to Optimization*. Optimization Software, Inc., New York, 1987.

J. Proakis. *Digital Communications*. McGraw-Hill, New York, fourth edition, 2001.

R. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence, 2008. URL `http://www.citebase.org/abstract?id=oai:arXiv.org:0811.3628`. Available from `arxiv.org/abs/0811.3628`.

A. J. Rothman, P. J. Bickel, E. Levina, and J. Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

R. Salvador, J. Suckling, C. Schwarzbauer, and E. Bullmore. Undirected graphs of frequency-dependent functional connectivity in whole brain networks. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1457):937–946, 2005.

K. Scheinberg and I. Rish. SINCO - a greedy coordinate ascent method for sparse inverse covariance selection problem. 2009. Submitted.

J. Songsiri, J. Dahl, and L. Vandenberghe. Graphical models of autoregressive processes. In Y. Eldar and D. Palomar, editors, *Convex Optimization in Signal Processing and Communications*. Cambridge University Press, Cambridge, 2009.

P. Stoica and R. L. Moses. *Introduction to Spectral Analysis*. Prentice Hall, London, 1997.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

J. Timmer, M. Lauk, S. Häußler, V. Radt, B. Köster, B. Hellwig, B. Guschlbauer, C.H. Lücking, M. Eichler, and G. Deuschl. Cross-spectral analysis of tremor time series. *Internation Journal of Bifurcation and Chaos in applied Sciences and Engineering*, 10(11):2595–2610, 2000.

K.-C. Toh. Primal-dual path-following algorithms for determinant maximization problems with linear matrix inequalities. *Computational Optimization and Applications*, 14:309–330, 1999.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *SIAM Journal on Optimization*, 2008. Submitted.

L. Vandenberghe, S. Boyd, and S.-P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM J. on Matrix Analysis and Applications*, 19(2):499–533, April 1998.

S.J. Wright, R.D. Nowak, and M.A.T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B Statistical Methodology*, 68(1):49–67, 2006.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19, 2007.

P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497, 2009.

Journal of Machine Learning Research 11 (2010) 2707-2744 Submitted 12/09; Revised 9/10; Published 10/10

Using Contextual Representations to Efficiently Learn Context-Free Languages

Alexander Clark ALEXC@CS.RHUL.AC.UK
Department of Computer Science,
Royal Holloway, University of London
Egham, Surrey, TW20 0EX
United Kingdom

Rémi Eyraud REMI.EYRAUD@LIF.UNIV-MRS.FR
Amaury Habrard AMAURY.HABRARD@LIF.UNIV-MRS.FR
Laboratoire d'Informatique Fondamentale de Marseille
CNRS UMR 6166, Aix-Marseille Université
39, rue Frédéric Joliot-Curie, 13453 Marseille cedex 13,
France

Editor: Fernando Pereira

Abstract

We present a polynomial update time algorithm for the inductive inference of a large class of context-free languages using the paradigm of positive data and a membership oracle. We achieve this result by moving to a novel representation, called Contextual Binary Feature Grammars (CBFGs), which are capable of representing richly structured context-free languages as well as some context sensitive languages. These representations explicitly model the lattice structure of the *distribution* of a set of substrings and can be inferred using a generalisation of distributional learning. This formalism is an attempt to bridge the gap between simple learnable classes and the sorts of highly expressive representations necessary for linguistic representation: it allows the learnability of a large class of context-free languages, that includes all regular languages and those context-free languages that satisfy two simple constraints. The formalism and the algorithm seem well suited to natural language and in particular to the modeling of first language acquisition. Preliminary experimental results confirm the effectiveness of this approach.

Keywords: grammatical inference, context-free language, positive data only, membership queries

1. Introduction

In natural language processing, many applications require the learning of powerful grammatical models. One of the central concerns of generative linguistics is the definition of an adequate formalism that needs to satisfy two different objectives. On the one hand, such a formalism must be expressive enough to describe natural languages. On the other hand, it has to be sufficiently constrained to be learnable from the sort of linguistic data available to the child learner (Chomsky, 1986). In this context, there are two possible research strategies. One is to take a descriptively adequate formalism such as *Tree Adjoining Grammars* (Joshi and Schabes, 1997) or some other mildly context sensitive grammatical formalism and try to construct learning algorithms for that class. However, such a strategy is unlikely to be successful because classes that are so powerful are difficult to handle from a machine learning point of view. The other approach, which we adopt

in this paper, consists in switching to a formalism that is in some sense intrinsically learnable, and seeing whether we can represent linguistically interesting formal languages in that representation.

Grammatical inference is the machine learning domain which aims at studying learnability of formal languages. While many learnability results have been obtained for regular languages (Angluin, 1987; Carrasco and Oncina, 1994), this class is not sufficient to correctly represent natural languages. The next class of languages to consider is the class of context-free languages (CFL). Unfortunately, there exists no learnability results for the whole class. This may be explained by the fact that this class relies on syntactic properties instead of intrinsic properties of the language like the notion of residuals for regular languages (Denis et al., 2004). Thus, most of the approaches proposed in the literature are either based on heuristics (Nakamura and Matsumoto, 2005; Langley and Stromsten, 2000) or are theoretically well founded but concern very restricted subclasses of context-free languages (Eyraud et al., 2007; Yokomori, 2003; Higuera and Oncina, 2002). Some of these approaches are built from the idea of *distributional learning*,[1] normally attributed to Harris (1954). The basic principle—as we reinterpret it in our work—is to look at the set of contexts that a substring can occur in. The distribution of a substring is the linguistic way of referring to this set of contexts. This idea has formed the basis of many heuristic algorithms for learning context-free grammars (see Adriaans, 2002 for instance). However, a recent approach by Clark and Eyraud (2007), has presented an accurate formalisation of distributional learning. From this formulation, a provably correct algorithm for context-free grammatical inference was given in the identification in the limit framework, albeit for a very limited subclass of languages, the substitutable languages. From a more general point of view, the central insight is that it is not necessary to find the non-terminals of the context-free grammar (CFG): it is enough to be able to represent the congruence classes of a sufficiently large set of substrings of the language and to be able to compute how they combine. This result was extended to a PAC-learning result under a number of different assumptions (Clark, 2006) for a larger class of languages, and also to a family of classes of learnable languages (Yoshinaka, 2008).

Despite their theoretical bases, these results are still too limited to form the basis for models for natural language. There are two significant limitations to this work: first it uses a very crude measure for determining the syntactic congruence, and secondly the number of congruence classes required will in real cases be prohibitively large. If each non-terminal corresponds to a single congruence class (the NTS languages Boasson and Senizergues, 1985), then the problem may be tractable. However in general the contexts of different non-terminals overlap enormously: for instance the contexts of adjective phrases and noun phrases in English both contain contexts of the form (*"it is"*, *"."*). Problems of lexical ambiguity also cause trouble. Thus for a CFG it may be the case that the number of congruence classes corresponding to each non-terminal may be exponentially large (in the size of the grammar). But the situation in natural language is even worse: the CFG itself may have an exponentially large number of non-terminals to start off with! Conventional CFGs are simply not sufficiently expressive to be cognitively plausible representations of natural language: to write a CFG requires a multiplication of the numbers of non-terminals to handle phenomena like *subject verb agreement*, *gender features*, *displaced constituents*, etc. This requires the use of a formalism like GPSG (Generalised Phrase Structure Grammar) (Gazdar et al., 1985) to write a meta-grammar—a compact way of specifying a very large CFG with richly structured non-terminals.

1. Note here that the word distributional does not refer to stochastic distributions, but to the occurrence of strings into contexts. The distribution of a string corresponds to all the possible contexts in which the string can appear.

Thus we cannot hope to learn natural languages by learning one congruence class at a time: it is vital to use a more structured representation.

This is the objective of the approach introduced in this article: for the first time, we can bridge the gap between theoretically well founded grammatical inference methods and the sorts of structured representations required for modeling natural languages.

In this paper, we present a family of representations for highly structured context-free languages and show how they can be learned. This is a paper in learning, but superficially it may appear to be a paper about grammatical representations: much of the work is done by switching to a more tractable formalism, a move which is familiar to many in machine learning. From a machine learning point of view, it is a commonplace that switching to a better representation—for example, through a non-linear map into some feature space—may make a hard problem very easy.

The contributions of this paper are as follows: we present in Section 3 a rich grammatical formalism, which we call *Contextual Binary Feature Grammars* (CBFG). This grammar formalism is defined using a set of contexts which play the role of features with a strict semantics attached to these features. Though not completely original, since it is closely related to a number of other formalisms such as Range Concatenation Grammars (Boullier, 2000), it is of independent interest. We consider then the case when the contextual features assigned to a string correspond to the contexts that the string can occur in, in the language defined by the grammar. When this property holds, we call it an *exact CBFG*. The crucial point here is that for languages that can be defined by an exact CBFG, the underlying structure of the representation relies on intrinsic properties of the language easily observable on samples by looking at context sets.

The learning algorithm is defined in Section 4. We provide some conditions, both on the context sets and the learning set, to ensure the learnability of languages that can be represented by CBFG. We prove that this algorithm can identify in the limit this restricted class of CBFGs from positive data and a membership oracle.

Some experiments are provided in Section 5: these experiments are intended to demonstrate that even quite naive algorithms based on this are efficient and effective at learning context-free languages.

Section 6 contains a theoretical study on the expressiveness of CBFG representations. We investigate the links with the classical Chomsky hierarchy, some well known grammatical representations used in natural language processing. An important result about the expressive power of the class of CBFG is obtained: it contains all the context-free languages and some non context-free languages. This makes this representation a good candidate for representing natural languages. However exact CBFG do not include all context-free languages but do include some non context-free ones, thus they are orthogonal with the classic Chomsky hierarchy and can represent a large class of languages. This expressiveness is strengthened by the fact that exact CBFG contains most of the existing learnable classes of languages.

2. Basic Definitions and Notations

We begin by some standard notations and definitions used all along the paper.

We consider a finite alphabet Σ as a finite non-empty set of symbols also called letters. A string (also called word) u over Σ is a finite sequence of letters $u = u_1 \cdots u_n$. Let $|u|$ denote the length of u. The set of all strings over Σ is denoted by Σ^*, corresponding to the free monoid generated by Σ. λ denotes the empty string and $\Sigma^+ = \Sigma^* \backslash \{\lambda\}$. A language L is any subset $L \subseteq \Sigma^*$.

We will write the concatenation of u and v as uv, and similarly for sets of strings. $u \in \Sigma^*$ is a substring of $v \in \Sigma^*$ if there are strings $l, r \in \Sigma^*$ such that $v = lur$. Define $Sub(u)$ to be the set of non-empty substrings of u. For a set of strings S define $Sub(S) = \bigcup_{u \in S} Sub(u)$.

A context is an element of $\Sigma^* \times \Sigma^*$. For a string u and a context $f = (l, r)$ we write $f \odot u = lur$; the insertion or wrapping operation. We extend this to sets of strings and contexts in the natural way. We define by $Con(w) = \{(l, r) | \exists u \in \Sigma^+ : lur = w\}$, that is, the set of all contexts of a word w. Similarly, for a set of strings, we define: $Con(S) = \bigcup_{w \in S} Con(w)$.

We give now a formal definition of the set of contexts since it represents a notion often used in the paper.

Definition 1 *The set of contexts, or context distribution, of a string u in a language L is, $C_L(u) = \{(l, r) \in \Sigma^* \times \Sigma^* | lur \in L\}$. We will often drop the subscript where there is no ambiguity.*

Definition 2 *Two strings u and v are syntactically congruent with respect to a language L, denoted $u \equiv_L v$, if and only if $C_L(u) = C_L(v)$.*

The equivalence classes under this relation are the *congruence* classes of the language.

After these basic definitions and notations, we recall here the definition of a context-free grammar which is a class which is close to the language class studied in this paper.

Definition 3 *A context-free grammar (CFG) is a quadruple $G = (\Sigma, V, P, S)$. Σ is a finite alphabet of terminal symbols, V is a set of non terminals s.t. $\Sigma \cap V = \emptyset$, $P \subseteq V \times (V \cup \Sigma)^+$ is a finite set of productions, $S \in V$ is the start symbol.*

We denote a production of P: $N \to \alpha$ with $N \in V$ and $\alpha \in (V \cup \Sigma)^+$. We will write $uNv \Rightarrow_G u\alpha v$ if there is a production $N \to \alpha$ in G. $\overset{*}{\Rightarrow}_G$ denotes the reflexive transitive closure of $\Rightarrow_G$.

The language defined by a CFG G is $L(G) = \{w \in \Sigma^* | S \overset{*}{\Rightarrow}_G w\}$. In the following, we will consider the CFG are represented in the *Chomsky normal form* (CNF), that is, with right hand side of production rules composed of exactly two non terminals or with exactly one terminal symbol.

In general we will assume that λ is not a member of any language.

3. Contextual Binary Feature Grammars (CBFG)

Distributional learning, in our view, involves explicitly modeling the distribution of the substrings of the language—we would like to model $C_L(w)$. Clearly a crucial element of this distribution is the empty context (λ, λ): $(\lambda, \lambda) \in C_L(w)$ if and only if $w \in L$. Our goal is to construct a representation that allows us to recursively compute a representation of the distribution of a string w, $C_L(w)$, from the (representations of) the distributions of its substrings.

The representation by *contextual binary feature grammars* relies on the inclusion relation between sets of contexts of language L. In order to introduce this formalism, we propose, for a start, to present some preliminary results on context inclusion. These results will lead us to define a relevant representation for modeling these inclusion dependencies by the notion of *contextual binary feature grammars*.

3.1 Preliminary Results about Context Inclusion

The objective of this section is to give some information about contexts that will help to give an intuition about the representation. The basic insight behind CBFGs is that there is a relation between

the contexts of a string w and the contexts of its substrings. This is given by the following trivial lemma:

Lemma 4 *For any language L and for any strings u, u', v, v' if $C(u) = C(u')$ and $C(v) = C(v')$, then $C(uv) = C(u'v')$.*

Proof We write out the proof completely as the ideas will be used later on. Suppose we have u, v, u', v' that satisfy the conditions. If $(l, r) \in C(uv)$, then $(l, vr) \in C(u)$ and thus $(l, vr) \in C(u')$. As a consequence, $(lu', r) \in C(v)$ and then $(lu', r) \in C(v')$ which implies that $(l, r) \in C(u'v')$. Symmetrically, by using the same arguments, we can show that $(l, r) \in C(u'v')$ implies (l, r) in $C(uv)$. Thus $C(uv) = C(u'v')$. ■

This establishes that the syntactic monoid $\Sigma^* / \equiv_L$ is well-defined; from a learnability point of view this means that if we want to compute the contexts of a string w we can look for a split into two strings uv where u is congruent to u' and v is congruent to v'; if we can do this and we know how u' and v' combine, then we know that the contexts of uv will be exactly the contexts of $u'v'$. There is also a slightly stronger result:

Lemma 5 *For any language L and for any strings u, u', v, v' if $C(u) \subseteq C(u')$ and $C(v) \subseteq C(v')$, then $C(uv) \subseteq C(u'v')$.*

Proof See proof of Lemma 4. ■

$C(u) \subseteq C(u')$ means that we can replace any occurrence in a sentence of u with a u', without affecting the grammaticality, but not necessarily vice versa. Note that none of these strings need to correspond to non-terminals: this is valid for any fragment of a sentence.

We will give a simplified example from English syntax: the pronoun "*it*" can occur everywhere that the pronoun "*him*" can, but not vice versa.[2] Thus given a sentence "*I gave him away*", we can substitute "*him*" for "*it*", to get the grammatical sentence "*I gave it away*", but we cannot reverse the process. For example, given the sentence "*it is raining*", we cannot substitute "*him*" for "*it*", as we will get the ungrammatical sentence "*him is raining*". Thus we observe $C(him) \subsetneq C(it)$.

Looking at Lemma 5 we can also say that, if we have some finite set of strings K, where we know the contexts, then:

Corollary 6 *For any language L and for any set of strings K, we have:*

$$C(w) \supseteq \bigcup_{\substack{u',v': \\ u'v'=w}} \bigcup_{\substack{u \in K: \\ C(u) \subseteq C(u')}} \bigcup_{\substack{v \in K: \\ C(v) \subseteq C(v')}} C(uv).$$

This is the basis of our representation: a word w is characterised by its set of contexts. We can compute the representation of w, from the representation of its parts u', v', by looking at all of the other matching strings u and v where we understand how they combine (with subset inclusion). Rather than representing just the congruence classes, we will represent the lattice structure of the set of contexts using subset inclusion; sometimes called Dobrušin-domination (Marcus, 1967).

The key relationships are given by context set inclusion. *Contextual binary feature grammars* allow a proper definition of the combination of context inclusion.

2. This example does not account for a number of syntactic and semantic phenomena, particularly the distribution of reflexive anaphors.

3.2 Contextual Binary Feature Grammars

The formalism of *contextual binary feature grammars* has some resemblance with *Generalized Phrase Structure Grammar (GPSG)* (Gazdar et al., 1985), and most importantly the class of *Range Concatenation Grammars (RCG)* (Boullier, 2000); these relationships will be detailed in Section 6. As we will see later, note that our formalism defines a class orthogonal to the class of context-free grammars, indeed the use of subsets inclusion allows to model non context-free languages.

Definition 7 *A Contextual Binary Feature Grammar (CBFG) G is a tuple* $\langle F, P, P_L, \Sigma \rangle$*:*

- F *is a finite set of contexts, (i.e.,* $F \subset \Sigma^* \times \Sigma^*$*) called features, where we write* $E = 2^F$ *for the power set of* F *defining the categories of the grammar, and where* $(\lambda, \lambda) \in F$*.*
- $P \subseteq E \times E \times E$ *is a finite set of productions that we write* $x \to yz$ *where* $x, y, z \in E$*,*
- $P_L \subseteq E \times \Sigma$ *is a set of lexical rules, written* $x \to a$*,*
- Σ *denotes the alphabet.*

Given a CBFG G we can recursively define a function f_G from $\Sigma^* \to E$ as follows:

$$
\begin{aligned}
& f_G(\lambda) = \emptyset, \\
& f_G(w) = \bigcup_{(c \to w) \in P_L} c && \text{iff } |w| = 1, \\
& f_G(w) = \bigcup_{u,v:uv=w} \; \bigcup_{\substack{x \to yz \in P: \\ y \subseteq f_G(u) \wedge \\ z \subseteq f_G(v)}} x && \text{iff } |w| > 1.
\end{aligned}
$$

Given a CBFG G and a string w it is possible to compute $f_G(w)$ in time $O(|F||P||w|^3)$ using standard dynamic programming techniques. A straightforward modification of the Cocke-Kasami-Younger algorithm for parsing Context-Free Grammars will suffice.

Thus a CBFG, G, defines for every string u a set of contexts $f_G(u)$: this will be a representation of the context distribution. $f_G(u)$ will be a subset of F: we will want $f_G(u)$ to approximate $C_L(u) \cap F$. The natural way to define the membership of a string w in $L(G)$ is to have the context $(\lambda, \lambda) \in f_G(w)$.

Definition 8 *The language defined by a CBFG G is the set of all strings that are assigned the empty context:* $L(G) = \{u | (\lambda, \lambda) \in f_G(u)\}$.

We give here more explanation about the function f_G. A rule $x \to yz$ is applied to analyse a string w if there is a split or *cut* of the string w into two strings u and v such that $uv = w$ s.t. $y \subseteq f_G(u)$ and $z \subseteq f_G(v)$—recall that y and z are sets of features.

One way of viewing a production $x \to yz$ is as an implication: if two strings u and v are such that they have the features y and z, then their concatenation will have the features x. As features correspond to contexts, intuitively, the relation given by the production rule is linked with Lemma 5: x is included in the set of features of $w = uv$. From this relationship, for any $(l, r) \in x$ we have $lwr \in L(G)$.

The complete computation of f_G is then justified by Corollary 6: $f_G(w)$ defines all the possible contextual features associated by G to w with all the possible cuts $uv = w$ (i.e., all the possible derivations).

Note the relation between the third clause above and Corollary 6. In general we will apply more than one production at each step of the analysis.

We will discuss the relation between this class and the class of CFGs in some detail in Section 6. For the moment, we will just make the following points. First, the representation is quite close to that of a CFG where the non-terminals correspond to sets of contexts (subsets of F). There are, however, crucial differences: the very fact that they are represented by sets means that the non-terminals are no longer atomic symbols but rather structures; the formalism can combine different rules together at each step. Secondly, the function f_G can combine different parsing paths. It is not the case that every feature assigned to w must come from the same split of w into u and v. Rather some features could come from one split, and some from another: these two sets of features can be combined in a single derivation even though they come from different splits (which correspond to different derivation trees in CFG). It is this property that takes the class of languages out of the class of context-free languages. In the special case where all of the productions involve only singleton sets then this will reduce to a CFG—the non-terminals will correspond to the individual features, and $f_G(w)$ will correspond to the set of non-terminals that can derive the string w.

Clearly by the definition of $L(G)$ we are forcing a correspondence between the occurrence of the context (λ,λ) in $C_L(w)$ and the occurrence of the feature (λ,λ) in $f_G(w)$. But ideally we can also require that f_G defines exactly the possible features that can be associated to a given string according to the underlying language. Indeed, we are interested in cases where there is a correspondence between the language theoretic interpretation of a context, and the occurrence of that context as a feature in the grammar: in this case the features will be observable which will lead to learnability.

This is formalised via the following definitions.

Definition 9 *Given a finite set of contexts $F = \{(l_1,r_1), \ldots, (l_n,r_n)\}$ and a language L we can define the context feature function $F_L : \Sigma^* \to 2^F$ which is just the function $u \mapsto \{(l,r) \in F | lur \in L\} = C_L(u) \cap F$.*

Using this definition, we now need a correspondence between the language theoretic context feature function F_L and the representation in our CBFG, f_G.

Definition 10 *A CBFG G is* exact *if for all $u \in \Sigma^*$, $f_G(u) = F_{L(G)}(u)$.*

Example. Let $L = \{a^n b^n | n > 0\}$. Let $\langle F, (\lambda,\lambda), P, P_L, \Sigma\rangle$ a CBFG s.t.

$$F = \{(\lambda,\lambda), (a,\lambda), (aab,\lambda), (\lambda,b), (\lambda,abb)\}.$$

The lexical productions in P_L are:

$$\{(\lambda,b), (\lambda,abb)\} \to a, and$$

$$\{(a,\lambda), (aab,\lambda)\} \to b.$$

Note that these lexical productions are of the form $x \to a$, where x is a subset of F, that is to say, a set of features. The rule $\{(\lambda,b), (\lambda,abb)\} \to a$ therefore says that the letter a will be assigned both

of the features/contexts (λ, b) and (λ, abb). Since this is the only lexical rule for a, we will have that $f_G(a) = \{(\lambda, b), (\lambda, abb)\}$. The productions in P, denoted by $x \to yz$, where x, y, z are again sets of contexts, are defined as:

$$\{(\lambda,\lambda)\} \to \{(\lambda,b)\}\{(aab,\lambda)\},$$
$$\{(\lambda,\lambda)\} \to \{(\lambda,abb)\}\{(a,\lambda)\},$$
$$\{(\lambda,b)\} \to \{(\lambda,abb)\}\{(\lambda,\lambda)\}, \textit{and}$$
$$\{(a,\lambda)\} \to \{(\lambda,\lambda)\}\{(aab,\lambda)\}.$$

In each of these rules, in this trivial case, the sets of contexts are singleton sets. In general, these productions may involve sets that have more than one element. This defines an exact CBFG for L. Indeed, the grammar assigns only (λ,λ) to the elements of the language; for all elements w of $\{a^n b^{n+1} : n > 1\}$ we have $f_G(w) = \{(a,\lambda)\} = F_L(w)$ and for all all elements w of $\{a^{n+1} b^n : n > 1\}$, $f_G(w) = \{(\lambda, b)\} = F_L(w)$; The lexical rules assign correct contexts to each letter.

Exact CBFGs are a more limited formalism than CBFGs themselves; without any limits on the interpretation of the features, we can define a class of formalisms that is equal to the class of *Conjunctive Grammars* (see Section 6.2.3). However, exactness is an important property because it allows to associate the intrinsic structure of a language to the structure of the representation. Contexts are easily observable from a sample and moreover it is only when the features correspond to the contexts that distributional learning algorithms can infer the structure of the language.

3.3 A Parsing Example

To clarify the relationship with CFG parsing, we will give a simple worked example. Consider the CBFG $G = \langle \{(\lambda,\lambda), (aab,\lambda), (\lambda,b), (\lambda,abb), (a,\lambda)\}, P, P_L, \{a,b\}\rangle$ with

$$P_L = \{ \quad \{(\lambda,b),(\lambda,abb)\} \to a, \quad \{(a,\lambda),(aab,\lambda)\} \to b \quad \}.$$

and

$$P = \{ \quad \{(\lambda,\lambda)\} \to \{(\lambda,b)\}\{(aab,\lambda)\}, \quad \{(\lambda,\lambda)\} \to \{(\lambda,abb)\}\{(a,\lambda)\}, \quad \{(\lambda,b)\} \to \{(\lambda,abb)\}\{(\lambda,\lambda)\}, \quad \{(a,\lambda)\} \to \{(\lambda,\lambda)\}\{(aab,\lambda)\} \quad \}.$$

If we want to parse a string w the usual way is to have a bottom-up approach. This means that we recursively compute the f_G function on the substrings of w in order to check whether (λ,λ) belongs to $f_G(w)$.

For example, suppose $w = aabb$. Figure 1 graphically gives the main steps of the computation of $f_G(aabb)$. Basically there are two ways to split $aabb$ that allow the derivation of the empty context: $aab|b$ and $a|abb$. The first one corresponds to the top part of the figure while the second one is drawn at the bottom. We can see for instance that the empty context belongs to $f_G(ab)$ thanks to the rule $\{(\lambda,\lambda)\} \to \{(\lambda,abb)\}\{(a,\lambda)\}$: $\{(\lambda,abb)\} \subseteq f_G(a)$ and $\{(a,\lambda)\} \subseteq f_G(b)$. But for symmetrical reasons the result can also be obtained using the rule $\{(\lambda,\lambda)\} \to \{(\lambda,b)\}\{(aab,\lambda)\}$.

As we trivially have $f_G(aa) = f_G(bb) = \emptyset$, since no right-hand side contains the concatenation of the same two features, an induction proof can be written to show that $(\lambda,\lambda) \in f_G(w) \Leftrightarrow w \in \{a^n b^n : n > 0\}$.

This is a simple example that illustrates the parsing of a string given a CBFG. This example does not fully express the power of CBFG since no element of the right hand side of a rule is composed of more than one context. A more complex example, corresponding to a context-sensitive language, will be presented in Section 6.1.3.

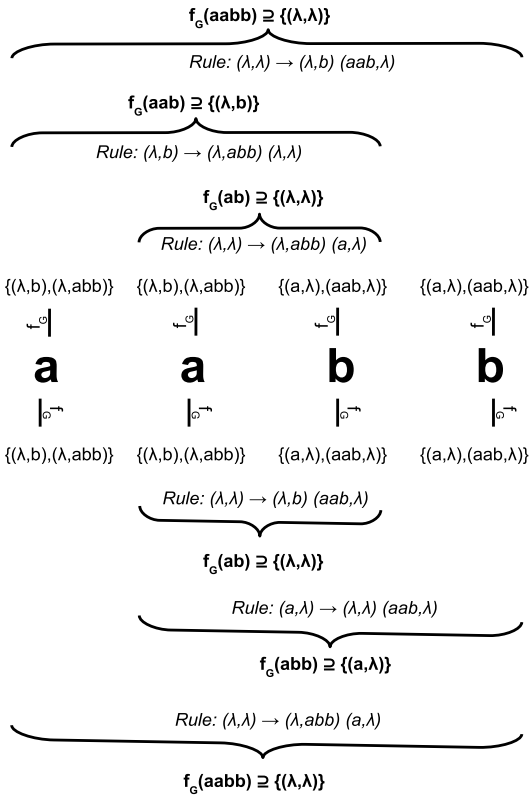


Figure 1: The two derivations to obtain (λ,λ) in $f_G(aabb)$ in the grammar G.

We stop here the presentation of the CBFG formalism and we present our learning algorithm in the next section. However, if the reader wishes to become more familiar with CBFGs a study on their expressiveness is provided in Section 6.

4. Learning Algorithm

We have carefully defined the representation so that the inference algorithm will be almost trivial. Given a set of strings, and a set of contexts, we can simply write down a CBFG that will approximate a particular language.

4.1 Building CBFGs from Sets of Strings and Contexts

Definition 11 *Let L be a language, F be a finite set of contexts such that $(\lambda,\lambda) \in F$, K a finite set of strings, $P_L = \{F_L(u) \to u | u \in K \wedge |u| = 1\}$ and $P = \{F_L(uv) \to F_L(u)F_L(v) | u, v, uv \in K\}$. We define $G_0(K,L,F)$ as the CBFG $\langle F, P, P_L, \Sigma\rangle$.*

Often K will be closed under substrings: that is, $Sub(K) = K$. This grammar is a CBFG, since K and F are finite, and so P and P_L are too by construction. In general it will not be exact.

We will call K here the *basis* for the language. The set of productions is defined merely by observation: we take the set of all productions that we observe as the concatenation of elements of the small set K.

Let us explain the construction in more detail. P_L is the set of lexical productions—analogous to rules of the form $N \to a$ in a CFG in Chomsky normal form. These rules just assign to the terminal symbols their observed distribution—this will obviously be correct in that $f_G(a) = F_L(a)$. P is the interesting set of productions: these allow us to predict the features of a string uv from the features of its part u and v. To construct P we take all triples of strings u, v, uv that are in our finite set K. We observe that u has the contexts $F_L(u)$ and v has the set of contexts $F_L(v)$: our rule then states that we can combine any string that has all of the contexts in $F_L(u)$ together with any string that has the contexts in $F_L(v)$ and the result will have all of the contexts in $F_L(uv)$.

We will now look at a simple example. Let $L = \{a^n b^n \mid n > 0\}$, F, the set of features is $\{(\lambda,\lambda),(a,\lambda),(\lambda,b)\}$ and K, the basis, is $\{a,b,ab,aa,aab\}$. For each of the elements of K we can compute the set of features that it has:

- $F_L(a)$ is just $\{(\lambda,b)\}$—this is the only one of the three contexts in F such that $f \odot a \in L$,
- $F_L(b) = \{(a,\lambda)\}$,
- $F_L(aa) = \emptyset$,
- $F_L(ab) = \{(\lambda,\lambda)\}$,
- $F_L(aab) = \{(\lambda,b)\}$.

G_0 will therefore have the following lexical productions $P_L = \{(\lambda,b)\} \to a,\ \{(a,\lambda)\} \to b$. We can now consider the productions in P. Looking at K we will see that there are only four possible triples of strings of the form uv, u, v: these are (aa,a,a), (ab,a,b), (aab,aa,b) and (aab,a,ab). Each of these will give rise to an element of P:

- The rule given by $ab = a \circ b$: $\{(\lambda,\lambda)\} \to \{(\lambda,b)\}\{(a,\lambda)\}$,
- $aa = a \circ a$ gives $\emptyset \to \{(\lambda,b)\}\{(\lambda,b)\}$,
- $aab = aa \circ b$ gives $\{(\lambda,b)\} \to \emptyset\{(a,\lambda)\}$,
- $aab = a \circ ab$ gives $\{(\lambda,b)\} \to \{(\lambda,b)\}\{(\lambda,\lambda)\}$.

Given K, F and an oracle for L we can thus simply write down a CBFG. However, in this case, the language $L(G_0)$ is not the same as L; moreover, the resulting grammar is not exact. Applying the rules for the recursive computation of f_G, we can see that $f_{G_0}(aab) = \{(\lambda,b)\}$ and $f_{G_0}(abb) = f_{G_0}(aabb) = \{(\lambda,b),(\lambda,\lambda)\}$ but $F_{L(G_0)}(abb) = \{(a,\lambda),(\lambda,b),(\lambda,\lambda)\}$ and thus G_0 is not exact. The problem here is caused by the fact that the production $\{(\lambda,b)\} \to \emptyset\{(a,\lambda)\}$ allows any string to occur in the place specified by the $\emptyset$: indeed since $\emptyset \subseteq f_{G_0}(aab)$ and $\{(a,\lambda)\} \subseteq f_{G_0}(b)$ the rule holds for $aabb$ and thus $\{(\lambda,b)\} \subseteq f_{G_0}(aabb)$. This is actually caused by the fact that there are no contexts in F that correspond to the string aa in K.

Fixing L for the moment, clearly the language defined depends on two factors: the set of strings K and the set of features F. Given K and F, and access to a membership oracle, we can write down

a CBFG with almost no computation, but we still have the problem of finding suitable K and F—it might be that searching for exactly the right combination of K and F is intractably hard. It turns out that it is also very easy to find suitable sets.

In the next section we will establish two important lemmas that show that the search for K and F is fundamentally tractable: first, that as K increases the language defined by $G_0(K,L,F)$ will increase, and secondly that as F increases the language will decrease.

Let us consider one example that illustrates these properties. Consider the language $L = \{a^n b \mid n \geq 0\} \cup \{ba^m \mid m \geq 0\} \cup \{a\}$.

First, let $K = \{a, b, ab\}$ and $F = \{(\lambda, \lambda)\}$; then, by the definition of G_0, we have the following productions:

- $\{(\lambda,\lambda)\} \to a$,
- $\{(\lambda,\lambda)\} \to b$,
- $\{(\lambda,\lambda)\} \to \{(\lambda,\lambda)\}\{(\lambda,\lambda)\}$.

It is easy to see that $L(G_0) = \Sigma^+$.

Now, suppose that $F = \{(\lambda,\lambda),(\lambda,b)\}$ with K unchanged; then, by construction G_0 will have the following productions:

- $\{(\lambda,\lambda),(\lambda,b)\} \to a$,
- $\{(\lambda,\lambda)\} \to b$,
- $\{(\lambda,\lambda)\} \to \{(\lambda,\lambda),(\lambda,b)\}\{(\lambda,\lambda)\}$.

The language defined by G_0 contains $a^n b$ and also a^n since $\{(\lambda,\lambda)\} \subset \{(\lambda,\lambda),(\lambda,b)\}$ allowing the third production to accept strings ending with an a. Thus, the language has been reduced such that $L(G_0) = \{a^n b \mid n \geq 0\} \cup \{a^m \mid m \geq 0\}$.

We continue by leaving $F = \{(\lambda,\lambda),(\lambda,b)\}$ and we enlarge K such that $K = \{a, b, ab, ba\}$. The productions in G_0 are:

- $\{(\lambda,\lambda),(\lambda,b)\} \to a$,
- $\{(\lambda,\lambda)\} \to b$,
- $\{(\lambda,\lambda)\} \to \{(\lambda,\lambda),(\lambda,b)\}\{(\lambda,\lambda)\}$; the rule given by $ab = a \circ b$,
- $\{(\lambda,\lambda)\} \to \{(\lambda,\lambda)\}\{(\lambda,\lambda),(\lambda,b)\}$; the rule given by $ba = b \circ a$.

The addition of the last rule allows the grammar to recognize ba^n and it can be easily shown that by a combination of the last two productions $a^n ba^m$ belongs to the language defined by the grammar. Then, $L(G_0)$ has been increased such that $L(G_0) = \{a^n ba^k \mid n,k \geq 0\} \cup \{a^m \mid m \geq 0\}$.

In this example, the addition of $(\lambda,b),(a,\lambda)$ and (λ,a) to F and the addition of aab and baa to K will then define the correct language. In fact this illustrates one principle of our approach: in the infinite data limit, the construction G_0 will define the correct language. In the following lemma we abuse notation and use G_0 for when we have infinite K, and F: in this lemma we let K be the set of all non-empty strings and we let F be the set of all possible contexts $(\Sigma^* \times \Sigma^*)$. Recall that in this case for every string w $C_L(w) = F_L(w)$.

Lemma 12 *For any language L, let $G = G_0(\Sigma^+, L, \Sigma^* \times \Sigma^*)$. Then for all $w \in \Sigma^+$ $f_G(w) = C_L(w)$ and therefore $L(G) = L$.*

Proof By induction on the length of w. If $|w| = 1$, and $w = a$ then there is a lexical production $C_L(a) \to a$ and by the definition of $f_G(a) = C_L(a)$. Suppose this is true for all w with $|w| \leq k$. Let w be some string of length $k+1$. Consider any split of w into u, v such that $w = uv$. $f_G(w)$ is the union over all these splits of a function. We will show that every such split will give the same result of $C_L(w)$. By inductive hypothesis $f_G(u) = C_L(u), f_G(v) = C_L(v)$. Since u, v, w are in $K = \Sigma^+$ we will also have an element of P of the form $C_L(w) \to C_L(u)C_L(v)$, so we know that $f_G(w) \supseteq F_L(w)$. We now show that f_G will not predict any extra contexts. Consider every production in P, $F_L(u'v') \to F_L(u')F_L(v')$, that applies to u, v, that is, with $F_L(u') \subseteq f_G(u) = C_L(u)$ and $F_L(v') \subseteq f_G(v) = C_L(v)$. Lemma 5 shows that in this case $F_L(u'v') \subseteq F_L(w)$ and thus we deduce that $f_G(w) \subseteq F_L(w)$, which establishes the lemma. ■

Informally if we take K to be every string and F to be every context, then we can accurately define any language. Of course, we are just interested in those cases where this can be defined finitely and we have a CBFG, in which case L will be decidable, but this infinite limit is a good check that the construction is sound.

4.2 Monotonicity Lemmas

We now prove two lemmas that show that the size of the language, and more particularly the features predicted will increase or decrease monotonically as a function of the basis K, and the feature set F, respectively. In fact, they give also a framework for approaching a target language from K and F.

Lemma 13 *Suppose we have two CBFGs defined by $G = G_0(K, L, F)$ and $G' = G_0(K, L, F')$ where $F \subseteq F'$. Then for all u, $f_G(u) \supseteq f_{G'}(u) \cap F$.*

Proof Let G' have a set of productions P', P'_L, and G have a set of productions P, P_L. Clearly if $x \to yz \in P'$ then $x \cap F \to (y \cap F)(z \cap F)$ is in P by the definition of G_0, and likewise for P_L, P'_L. By induction on $|u|$ we can show that any feature in $f_{G'}(u) \cap F$ will be in $f_G(u)$. The base case is trivial since $F'_L(a) \cap F = F_L(a)$; if it is true for all strings up to length k, then if $f \in f_{G'}(u) \cap F$; there must be a production in F' with f on the head. By the inductive hypothesis, the right hand sides of the corresponding production in P will be triggered, and so f must be in $f_G(u)$. ■

Corollary 14 *Suppose we have two CBFGs defined by $G = G_0(K, L, F)$ and $G' = G_0(K, L, F')$ where $F \subseteq F'$; then $L(G) \supseteq L(G')$.*

Proof It is sufficient to remark that if $u \in L(G')$ then $(\lambda, \lambda) \in f_{G'}(u) \subseteq f_G(u)$ and thus $u \in L(G)$. ■

Conversely, we can show that as we increase K, the language and the map f_G will increase. This is addressed by the next lemma.

Lemma 15 *Suppose we have two CBFGs defined by $G = G_0(K, L, F)$ and $G' = G_0(K', L, F)$ where $K \subseteq K'$. Then for all u, $f_{G_0(K,L,F)}(u) \subseteq f_{G_0(K',L,F)}(u)$.*

Proof Clearly the sets of productions of $G_0(K,L,F)$ will be a subset of the set of productions of $G_0(K',L,F)$, and so anything that can be derived by the first can be derived by the second, again by induction on the length of the string. ■

A simple result is that when K contains all of the substrings of a word, then $G_0(K,L,F)$ will generate all of the correct features for this word.

Lemma 16 *For any string w, if $Sub(w) \subset K$, and let $G = G_0(K,L,F)$, then $F_L(w) \subseteq f_G(w)$.*

Proof By recursion on the size of w. Let $G = G_0(K,L,F) = \langle F, P, P_L, \Sigma \rangle$. First, notice that if w is of length 1 then we have $F_L(w) \to w$ in P_L and thus the lemma holds. Then suppose that $|w| = k \geq 2$. Let u and v in Σ^+ be such that $w = uv$. As $Sub(w) \subset K$ we have u, v in K. Therefore the rule $F_L(w) \to F_L(u)F_L(v)$ belongs to P. As $|u| < |w|$ and $|v| < |w|$, by recursion we get $F_L(u) \subseteq f_G(u)$ and $F_L(v) \subseteq f_G(v)$. Thus the rule can be applied and then $F_L(w) \subseteq f_G(w)$. ■

In particular if $w \in L$, and $Sub(w) \subseteq K$, then $w \in L(G)$. This means that we can easily increase the language defined by G just by adding $Sub(w)$ to K. In general we do not need to add every element of $Sub(w)$—it is enough to have one binary bracketing.

To establish learnability, we need to prove that for a target language L, if we have a sufficiently large F then $L(G_0(K,L,F))$ will be contained within L and that if we have a sufficiently large K, then $L(G_0(K,L,F))$ will contain L.

4.3 Fiducial Feature Sets and Finite Context Property

We need to be able to prove that for any K if we have enough features then the language defined will be included within the target language L. We formalise the idea of having enough features in the following way:

Definition 17 *For a language L and a string u, a set of features F is fiducial on u if for all $v \in \Sigma^*$, $F_L(u) \subseteq F_L(v)$ implies $C_L(u) \subseteq C_L(v)$.*

Note that if F is fiducial on u and $F \subset F'$ then F' is fiducial on u. Therefore we can naturally extend this to sets of strings.

Definition 18 *For a language L and a set of strings K, a set of features F is fiducial on K if for all $u \in K$, F is fiducial on u.*

Clearly, for any string w, $C_L(w)$ will be fiducial on w; but this is vacuous—we are interested in cases where there is a finite set of contexts which is fiducial for w, but where $C_L(w)$ is infinite. If u and v are both in K then having the same features means they are syntactically congruent. However if two strings, neither of which are in K, have the same features this does not mean they are necessarily congruent (for instance if $F_L(v) = F_L(v') = \emptyset$). For non finite state languages, the set of congruence classes will be infinite, and thus we cannot have a finite fiducial set for the set of all strings in $Sub(L)$, but we can have a feature set that is correct for a finite subset of strings, or more generally for an infinite set of strings, if they fall into a finite number of congruence classes.

Let us consider our running example $L = \{a^n b^n | n > 0\}$. Take the string ab. $C_L(ab)$ is infinite and contains contexts of the form $(\lambda,\lambda),(a,b),(aa,bb)$ and so on. Consider a set with just one of these contexts, say $F = \{(a,b)\}$. This set is clearly fiducial for ab, since the only strings that will have this context are those that are congruent to ab. Consider now the string aab; clearly $\{(\lambda,b)\}$ is fiducial for aab, even though the string a, which is not congruent to aab also occurs in this context. Indeed, this does not violate fiduciality since $C_L(a) \supset C_L(aab)$. However, looking at string a, $\{(\lambda,b)\}$ is not fiducial, since aab has this context but does not include all the contexts of a such as, for example, (λ,abb).

In these trivial examples, a context set of cardinality one is sufficient to be fiducial, but this is not the case in general. Consider the finite language $L = \{ab,ac,db,ec,dx,ey\}$, and the string a. It has two contexts (λ,b) and (λ,c) neither of which is fiducial for a on its own. However, the set of both contexts is: $\{(\lambda,b),(\lambda,c)\}$ is fiducial for a.

We now define the finite context property, which is one of the two conditions that languages must satisfy to be learnable in this model; this condition is a purely language theoretic property.

Definition 19 *A language L has the Finite Context Property (FCP) if every string has a finite fiducial feature set.*

Clearly if L has the FCP, then any finite set of substrings, K, has a finite fiducial feature set which will be the union of the finite fiducial feature sets for each element of K. If $u \notin Sub(L)$ then any set of features is fiducial since $C_L(u) = \emptyset$.

We note here that all regular languages have the FCP. We refer the reader to the Section 6.1.1 about CBFG and regular languages where the Lemma 35 and the associated construction proves this claim.

We can now state the most important lemma: this lemma links up the definition of the feature map in a CBFG, with the fiducial set of features to show that only correct features will be assigned to substrings by the grammar. It states that the features assigned by the grammar will correspond to the language theoretic interpretation of them as contexts.

Lemma 20 *For any language L, given a set of strings K and a set of features F, let $G = G_0(K,L,F)$. If F is fiducial on K, then for all $w \in \Sigma^*$ $f_G(w) \subseteq F_L(w)$.*

Proof We proceed by induction on length of the string. *Base case:* strings of length 1. $f_G(w)$ will be the set of observed contexts of w, and since we have observed these contexts, they must be in the language. *Inductive step:* let w a string of length k.

Take a feature f on $f_G(w)$; by definition this must come from some production $x \to yz$ and a split u,v of w. The production must be from some elements of K, u',v' and $u'v'$ such that $y = F_L(u'), z = F_L(v')$ and $x = F_L(u'v')$. If the production applies this means that $F_L(u') = y \subseteq f_G(u) \subseteq F_L(u)$ (by inductive hypothesis), and similarly $F_L(v') \subseteq F_L(v)$. By fiduciality of F this means that $C(u') \subseteq C(u)$ and $C(v') \subseteq C(v)$. So by Lemma 5 $C(u'v') \subseteq C(uv)$. Since $f \in C(u'v')$ then $f \in C(uv) = C(w)$. Therefore, since $f \in F$ and $C(w) \cap F = F_L(w)$, $f \in F_L(w)$, and therefore $f_G(w) \subseteq F_L(w)$. ■

Corollary 21 *If F is fiducial on K then $L(G_0(K,F,L)) \subseteq L$.*

Therefore for any finite set K from an FCP language, we can find a set of features so that the language defined by those features on K is not too big.

4.4 Kernel and Finite Kernel Property

We will now show a complementary result, namely that for a sufficiently large K the language defined by G_0 will include the target language. We will start by formalising the idea that a set K is large enough, by defining the idea of a *kernel*.

Definition 22 *A finite set $K \subseteq \Sigma^*$ is a kernel for a language L, if for any set of features F, $L(G_0(K,F,L)) \supseteq L$.*

Consider again the language $L = \{a^n b^n | n \geq 0\}$. The set $K = \{a, b, ab\}$ is not a kernel, since if we have a large enough set of features, then the language defined will only be $\{ab\}$ which is a proper subset of L. However $K = \{a, b, ab, aab, abb, aabb\}$ is a kernel: no matter how large a set of features we have the language defined will always include L. Consider a language $L' = L \cup \{b^{16}\}$. In this case, a kernel for L' must include, as well as a kernel for L, some set of substrings of b^{16}: it is enough to have b^{16}, b^8, b^4, bb, b.

To prove that a set is a kernel, it suffices to show that if we consider all the possible features for building the grammar, we will contain the target language; any smaller set of features defines then a larger language. In our case, we can take the infinite set of all contexts and define productions based on the congruence classes. If F is the set of all contexts then we have $F_L(u) = C_L(u)$, thus the productions will be exactly of the form $C(uv) \to C(u)C(v)$. This is a slight abuse of notation since feature sets are normally finite.

Lemma 23 *Let $F = \Sigma^* \times \Sigma^*$; if $L(G_0(K,L,F)) \supseteq L$ then K is a kernel.*

Proof By monotonicity of F: any finite feature set will be a subset of F. ■

Not all context-free languages will have a finite kernel. For example $L = \{a^+\} \cup \{a^n b^m | n < m\}$ is clearly context-free, but does not have a finite kernel. Assume that the set K contains all strings of length less than or equal to k. Assume w.l.o.g. that the fiducial set of features for K includes all features (λ, b^i), where $i \leq k+1$. Consider the rules of the form $F_L(a^k) \to F_L(a^j)F_L(a^{k-j})$; we can see that no matter how large k is, the derived CBFG will undergenerate as a^k is not congruent to a^{k-1}.

Definition 24 *A context-free grammar $G_T = \langle V, S, P, \Sigma \rangle$ has the Finite Kernel Property (FKP) iff for every non-terminal $N \in V$ there is a finite set of strings $K(N)$ such that $a \in K(N)$ if $a \in \Sigma$ and $N \to a \in P$ and such that for all $k \in K(N), N \stackrel{*}{\Rightarrow} k$ and where for every string $w \in \Sigma^*$ such that $N \stackrel{*}{\Rightarrow} w$ there is a string $k \in K(N)$ such that $C(k) \subseteq C(w)$. A CFL L has the FKP, if there is a grammar in CNF for it with the FKP.*

Notice that all regular languages have the FKP since they have a finite number of congruence classes.

Lemma 25 *Any context-free language with the FKP has a finite kernel.*

Proof Let $G_T = \langle V, S, P, \Sigma \rangle$ be such a CNF CFG with the FKP. Define

$$K(G_T) = \bigcup_{N \in V} \left(K(N) \cup \bigcup_{X \to MN \in P} K(M)K(N) \right).$$

We claim that $K(G_T)$ is a kernel. Assume that $F = \Sigma^* \times \Sigma^*$ and let G be such that $G = G_0(K(G_T), L(G_T), F) = \langle F, (\lambda,\lambda), P, P_L, \Sigma \rangle$.

We will show, by induction on the length of derivation of w in G_T, that for all N, w if $N \stackrel{*}{\Rightarrow} w$ then there is a k in $K(N)$ such that $f_G(w) \supseteq C(k)$. If length of derivation is 1, then this is true since $|w| = 1$ and thus $w \in K(N)$: therefore $C(w) \to w \in P_L$. Suppose it is true for all derivations of length less than j. Take a derivation of length j; say $N \stackrel{*}{\Rightarrow} w$. There must be a production in G_T of the form $N \to PQ$, where $P \Rightarrow^* u$ and $Q \Rightarrow^* v$, and $w = uv$. By inductive hypothesis; we have $f_G(u) \supseteq C(k_u)$ and $f_G(v) \supseteq C(k_v)$. By construction $k_u k_v \in K(G_T)$ and then there will be a rule $C(k_u k_v) \to C(k_u)C(k_v)$ in P. Therefore $f_G(uv) \supseteq C(k_u k_v)$. Since $N \stackrel{*}{\Rightarrow} k_u k_v$ there must be some $k_{uv} \in K(N)$ such that $C(k_{uv}) \subseteq C(k_u k_v)$. Therefore $f_G(w) \supseteq C(k_u k_v) \supseteq C(k_{uv})$.

Now we can see that if $w \in L$, then $S \stackrel{*}{\Rightarrow} w$, then there is a $k \in K(S)$ such that $f_G(w) \supseteq C(k)$ and $S \stackrel{*}{\Rightarrow} k$, therefore $(\lambda,\lambda) \in f_G(w)$ since $(\lambda,\lambda) \in C(k)$, thus $w \in L(G)$ and therefore K is a kernel. ■

4.5 Learning Algorithm

Before we present the algorithm, we will discuss the learning model that we use. The class of languages that we will learn is suprafinite and thus we cannot get a positive data only identification in the limit result (Gold, 1967). Ultimately we are interested in a more realistic probabilistic learning paradigm, but for mathematical convenience it is appropriate to establish the basic results in a symbolic paradigm. The ultimate goal is to model natural languages, where negative data, or equivalence queries are generally not available or are computationally impossible. Accordingly, we have decided to use the model of positive data together with membership queries: an oracle can tell the learner whether a string is in the language or not (Angluin, 1988). The presented algorithm runs in time polynomial in the size of the sample S: since the strings are of variable length, this size must be the sum of the lengths of the strings in S, $\|S\| = \sum_{w \in S} |w|$. We should note that this is not a strong enough result: Pitt (1989) showed that any algorithm can be made polynomial, by only processing a small prefix of the data. It is hard to tighten the model sufficiently: the suggestion of de la Higuera (1997) for a polynomial characteristic set is inapplicable for representations, such as the ones in this paper, that are powerful enough to define languages whose shortest strings are exponentially long. Accordingly we do not require in this model a polynomial dependence on the size of the representation. We note that the situation is unsatisfactory, but we do not intend to propose a solution in this paper. We merely point out that the algorithm is genuinely polynomial and processes all of the data in the sample without "delaying tricks" of the type discussed by Pitt.

Definition 26 *A class of languages $\mathbb{L}$ is identifiable in the limit (IIL) from positive data and a membership oracle with polynomial time and queries iff there exist two polynomials $p(), q()$ and an algorithm A such that:*

- *Given an infinite presentation of positive examples S, where S_n is the first n examples of the presentation,*
  1. *A returns a representation $G = A(S_n)$ in time $p(\|S_n\|)$.*
  2. *A asks at most $q(\|S_n\|)$ queries to build $A(S_n)$.*
- *For each language $L \in \mathbb{L}$, for each presentation S of L, there exists an index n such that for all $N \geq n$: $A(S_N) = A(S_n)$ and $L(A(S_n)) = L$.*

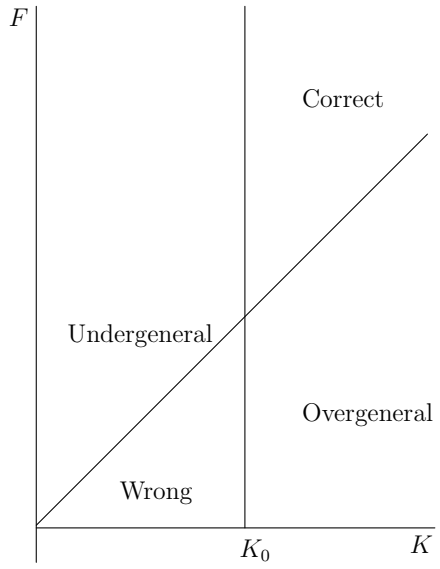


Figure 2: The relationship between K and F: The diagonal line is the line of fiduciality: above this line means that F is fiducial on K. K_0 is a kernel for the language.

Before we present the algorithm we hope that it is intuitively obvious how the approach will work. Figure 2 diagrammatically shows the relationship between K and F. When we have a large enough K, we will be to the right of the vertical line; when we have enough features for that K we will be above the diagonal line. Thus the basis of the algorithm is to move to the right, until we have enough data, and then to move up vertically, increasing the feature set until we have a fiducial set.

We can now define our learning algorithm in Algorithm 1. Informally, D is the list of all strings that have been seen so far and G_n is the current grammar obtained with the first n strings of D. The algorithm uses two tests: one test is just to determine if the current hypothesis undergeneralises. This is trivial, since we have a positive presentation of the data, and so eventually we will be presented with a string in $L \setminus L(G_n)$. In this case we need to increase K; we accordingly increase K to the set of all substrings that we have observed so far. The second test is a bit more delicate. We want to detect if our algorithm overgeneralises. This requires us to search through a polynomially bounded set of strings looking for a string that is in $L(G_n) \setminus L$. An obvious candidate set is $Con(D) \odot Sub(D)$; but though we conjecture that this is adequate, we have not yet been able to prove that is correct, as it might be that the overgenerated string does not lie in $Con(L) \odot Sub(L)$.

Here we use a slightly stricter criterion: we try to detect whether F is fiducial for K: we search through a polynomially bounded set of strings, $Sub(D)$, to find a violation of the fiduciality condition. If we find such a violation, then we know that F is not fiducial for K, and so we increase F to the set of all contexts that we have seen so far, $Con(D)$.

In Algorithm 1, $G_0(K, O, F)$ denotes the same construction as $G_0(K, L, F)$, except that we use membership queries with the oracle O to compute F_L for each element in K. We give the identification in the limit version of the algorithm, that is, that admits an infinite positive presentation of strings in input.

Algorithm 1: CBFG learning algorithm IIL

Data: A sequence of strings $S = \{w_1, w_2 \ldots, \}$, membership oracle O
Result: A sequence of CBFGs $G_1, G_2, \ldots$
$K \leftarrow \emptyset$; $D \leftarrow \emptyset$; $F \leftarrow \{(\lambda, \lambda)\}$; $G = G_0(K, O, F)$;
for w_i **do**
  $D \leftarrow D \cup \{w_i\}$; $C \leftarrow Con(D)$; $S \leftarrow Sub(D)$;
  if $\exists w \in D \setminus L(G)$ **then**
    $K \leftarrow S$; $F \leftarrow C$;
  end
  else if $\exists v \in S, u \in K, f \in C$ *such that* $F_L(u) \subseteq F_L(v)$ *and* $f \odot u \in L$ *but* $f \odot v \notin L$ **then**
    $F \leftarrow C$;
  end
  $G = G_0(K, O, F)$;
  Output $G_i = G$;
end

Theorem 27 *Algorithm 1 runs in polynomial time in the size of the sample, and makes a polynomial number of calls to the membership oracle.*

Proof The value of D will just be the set of observed strings; $Sub(D)$ and $Con(D)$ are both polynomially bounded by the size of the sample, and therefore so are $|K|$ and $|F|$. Therefore the number of calls to the oracle is clearly polynomial, as it is bouned by $|K||F|$. Computing G_0 is also polynomial, since $|P| \leq |K|^2$, and all strings involved are in $Sub(D)$. ■

4.6 Identification in the Limit Result

In the following, we consider the class of context-free languages having the FCP and the FKP, represented by CBFG. K_n denotes the value of K at the n^{th} loop, and similarly for F, D.

Definition 28 *$\mathcal{L}_{CFG}$ is the class of all context-free languages that satisfy the FCP and the FKP.*

In what follows we assume that L is an element of this class, and that $w_1, \ldots, w_n, \ldots$ is a infinite presentation of the language. The proof is straightforward and merely requires an analysis of a few cases. We will proceed as follows: there are 4 states that the model can be in, that correspond to the four regions of the diagram in Figure 2.

1. K is a kernel and F is fiducial for K; in this case the model has converged to the correct answer. This is the region labeled *correct* in Figure 2.
2. K is a kernel and F is not fiducial for K: then $L \subseteq L(G)$, and at some later point, we will increase F to a fiducial set, and we will be in state 1: this is the region labeled *overgeneral*.
3. K is not a kernel and F is fiducial. Either $L(G) = L$, in which case we have converged to a correct answer or, if not, we will define a proper subset of the target language. In the later case we will change hypothesis at some later point, increase K to a kernel, and move to state 2 or state 1. This is the area labeled *undergeneral*.

4. K is not a kernel and F is not fiducial: in this case at some point we will move to states 1 or 2. This is the area labeled *wrong*.

We will start by making some basic statements about properties of the algorithm:

Lemma 29 *If there is some n, such that F_n is fiducial for K_n and $L(G_n) = L$, then the algorithm will not change its hypothesis: that is, for all $n > N$, $K_n = K_N, F_n = F_N$ and therefore $G_n = G_N$.*

Proof If $L(G_n)$ is correct, then the first condition of the loop will never be met; if F_n is fiducial for K_n, then the second condition will never be satisfied. ■

Lemma 30 *If there is some N such that K_N is a kernel, then for all $n > N$, $K_n = K_N$.*

Proof Immediate by definition of a kernel, and of the algorithm. ■

We now prove that if F is not fiducial then the algorithm will be able to detect this.

Lemma 31 *If there is some n such that F_n is not fiducial for K_n, then there is some index $n' \geq n$ at which F_n will be increased.*

Proof If F_n is not fiducial, then by definition there is some $u \in K$, $v \in \Sigma^+$ such that $F_L(u) \subseteq F_L(v)$, but there is an $f \in C_L(u)$ that is not in $C_L(v)$. By construction $F_L(u)$ is always non-empty, and so is $F_L(v)$. Thus $v \in Sub(L)$. Note $f \odot u \in L$, so $f \in Con(L)$. Let n' be the smallest index such that $v \in Sub(D_n)$ and $f \in Con(D_n)$: at this point, either F_n will have changed, or not, in which case it will be increased at this point. ■

We now prove that we will always get a fiducial feature set.

Lemma 32 *For any n, there is some n' such that $F_{n'}$ is fiducial for K_n.*

Proof If F_n is fiducial then $n' = n$ satisfies the condition. Assume otherwise. Let F be a finite set of contexts that is fiducial for K_n. We can assume that $F \subseteq Con(L)$. Let n_1 be the first index such that $Con(D_{n_1})$ contains F. At this point we are not sure that $F_{n_1} = Con(D_{n_1})$ since the conditions for changing the set of contexts may not be reached. Anyhow, if it is the case then F_{n_1} is fiducial, then $n_1 = n'$ satisfies the condition. If not, then by the preceding lemma, there must be some point n_2 at which we will increase the set of contexts of the current grammar; $F_{n_2} = Con(n_2)$ must contain F since $Con(D_{n_1}) \subset Con(D_{n_2})$, and is therefore fiducial, and so $n_2 = n'$ satisfies the condition. ■

Lemma 33 *For every positive presentation of an $L \in \mathcal{L}_{CFG}$, there is some n such that either the algorithm has converged to a correct grammar or K_n is a kernel.*

Proof Let m be the smallest number such that $Sub(D_m)$ is a kernel. Recall that any superset of a kernel is a kernel, and that all CFL with the FKP have a finite kernel (Lemma 25), and that such a kernel is a subset of $Sub(L)$, so such an m must exist.

Consider the grammar G_m; there are three possibilities:
1. $L(G_m) = L$, and F_m is fiducial, in which case the grammar has converged.
2. $L(G_m)$ is a proper subset of L and F_m is fiducial. Let m' be the first point at which $w_{m'}$ is in $L \setminus L(G_m)$; at this point $K_{m'}$ will be increased to include $Sub(D_m)$ and it will therefore be a kernel.
3. F_m is not fiducial: in this case by Lemma 32; there is some n at which F_n is fiducial for K_m. Either $K_n = K_m$ in which case this reduces to Case 2; or K_n is larger than K_m in which case it must be a kernel, since it will include $Sub(D_m)$ which is a kernel.

■

We now can prove the main result of the paper:

Theorem 34 *Algorithm 1 identifies in the limit the class of context-free languages with the finite context property and the finite kernel property.*

Proof By Lemma 33 there is some point at which it converges or has a kernel. If K_n is a kernel then by Lemma 32, there is some point n' at which we have a fiducial feature set. Therefore $L(G_{n'}) = L$, and the algorithm has converged. ■

4.7 Examples

We will now give a worked example of the algorithm.

Suppose $L = \{a^n b^n | n > 0\}$.

G_0 will be the empty grammar, with $K = \emptyset, F = \{(\lambda, \lambda)\}$ and an empty set of productions. $L(G_0) = \emptyset$.

1. Suppose $w_1 = ab$. $D = \{ab\}$. This is not in $L(G_0)$ so we set

   - $K = Sub(D) = \{a, b, ab\}$.
   - $F = Con(D) = \{(\lambda, \lambda), (a, \lambda), (\lambda, b)\}$.

   This gives us one production: $F_L(ab) \to F_L(a)F_L(b)$ which corresponds to $\{(\lambda, \lambda)\} \to \{(\lambda, b)\}\{(a, \lambda)\}$, and the lexical productions $\{(\lambda, b)\} \to a, \{(a, \lambda)\} \to b$. The language defined is thus $L(G_1) = \{ab\}$.

2. Suppose $w_2 = aabb$. $D = \{ab, aabb\}$. This is not in $L(G_1)$ so we set

   - $K = Sub(D) = \{a, b, ab, aa, bb, aab, abb, aabb\}$.
   - $F = Con(D) = \{(\lambda, \lambda), (a, \lambda), (\lambda, b), (aa, \lambda), (a, b), (\lambda, bb), (aab, \lambda), (aa, b), (a, bb), (\lambda, abb)\}$.

   We then have the following productions:

- $F_L(ab) \to F_L(a), F_L(b)$ which is
$$\{(\lambda,\lambda),(a,b)\} \to \{(a,bb),(\lambda,abb),(\lambda,b)\},\{(aa,b),(aab,\lambda),(a,\lambda)\}.$$
- $F_L(aab) \to F_L(a), F_L(ab)$ which is
$$\{(a,bb),(\lambda,b)\} \to \{(a,bb),(\lambda,abb),(\lambda,b)\},\{(\lambda,\lambda),(a,b)\}.$$
- $F_L(aab) \to F_L(aa), F_L(b)$ which is
$$\{(a,bb),(\lambda,b)\} \to \{(\lambda,bb)\},\{(aa,b),(aab,\lambda),(a,\lambda)\}.$$
- $F_L(bb) \to F_L(b), F_L(b)$ which is
$$\{(aa,\lambda)\} \to \{(aa,b),(aab,\lambda),(a,\lambda)\},\{(aa,b),(aab,\lambda),(a,\lambda)\}.$$
- $F_L(aa) \to F_L(a), F_L(a)$ which is
$$\{(\lambda,bb)\} \to \{(a,bb),(\lambda,abb),(\lambda,b)\},\{(a,bb),(\lambda,abb),(\lambda,b)\}.$$
- $F_L(aabb) \to F_L(a), F_L(abb)$ which is
$$\{(\lambda,\lambda),(a,b)\} \to \{(a,bb),(\lambda,abb),(\lambda,b)\},\{(aa,b),(a,\lambda)\}.$$
- $F_L(aabb) \to F_L(aa), F_L(bb)$ which is
$$\{(\lambda,\lambda),(a,b)\} \to \{(\lambda,bb)\},\{(aa,\lambda)\}.$$
- $F_L(aabb) \to F_L(aab), F_L(b)$ which is
$$\{(\lambda,\lambda),(a,b)\} \to \{(a,bb),(\lambda,b)\},\{(aa,b),(aab,\lambda),(a,\lambda)\}.$$
- $F_L(abb) \to F_L(a), F_L(bb)$ which is
$$\{(aa,b),(a,\lambda)\} \to \{(a,bb),(\lambda,abb),(\lambda,b)\},\{(aa,\lambda)\}.$$
- $F_L(abb) \to F_L(ab), F_L(b)$ which is
$$\{(aa,b),(a,\lambda)\} \to \{(\lambda,\lambda),(a,b)\},\{(aa,b),(aab,\lambda),(a,\lambda)\}.$$

and the two lexical productions:

- $F_L(a) \to a$ which is $\{(a,bb),(\lambda,abb),(\lambda,b)\} \to a$
- $F_L(b) \to b$ which is $\{(aa,b),(aab,\lambda),(a,\lambda)\} \to b$.

K is now a kernel and $L(G) = L$, but F is not fiducial for K, since (λ,bb) is not fiducial for aa (consider $aaab$).

3. Suppose $w_3 = aaabbb$. Now $|Con(D_3)| = 21$; there are now several elements of $Con(D_3)$ that are similar. For example $(\lambda,\lambda),(a,b)$ and (aa,bb) are identical but as it is harmless for the resulting grammar, it does not mind. Now we will detect that F is not fiducial: we will find $v = aaab$, $u = aa$ and $f = (\lambda,abbb)$; $F_L(aa) = \{(\lambda,bb)\} = F_L(aaab)$, but $f \odot aaab = aaababbb$ which is not in L. We will therefore increase F to be $Con(D_3)$, and then the algorithm will have converged. The final grammar will have 10 productions and 2 lexical productions; $|K| = 8$ and $|F| = 21$.

5. Practical Behavior of the Algorithm

In this section, we propose to study the behavior of our algorithm from a practical point of view. We focus more specifically on two important issues. The first one deals with the learning ability of the algorithm when the conditions for the theoretical learning result are not reached. Indeed, although the identification in the limit paradigm proves that with sufficient data it is possible to obtain exact convergence, it says nothing about the convergence when fewer learning examples are available: does the output get closer and closer to the target until it reaches it or does it stay far from the expected solution until receives enough data? The second question concerns the learning behavior of the algorithm: does it tend to over-generalise or to under-generalise?

For our experimental setup, we need to select appropriate data sets. In grammatical inference little has been done concerning benchmarking. The main available corpora are those of the on line competitions organised by the International Colloquium on Grammatical Inference. Three different competitions have recently taken place: the *Abbadingo One* (Lang et al., 1998) which was about regular languages, the *Omphalos* competition on context-free languages (Starkie et al., 2004) and the *Tenjino* competition (Starkie et al., 2006) dealing with transducers learning. Note that some of these data sets correspond to extremely hard learning problems since their main objective was to push the state of the art (some problems of the *Abbadingo One* competition are still unsolved more than ten years after its official end!)

However, these data sets can not be directly used for evaluating our algorithm because the solutions or the target models are not available. Our algorithm needs an oracle and thus we need a way to give answers to membership queries. In order to overcome this drawback, we chose to build synthetically some data sets following the experimental setup proposed by these competitions. More precisely, we decided to randomly generate target context-free grammars following what has been done for the *Omphalos* competition. Each grammar is then used either to generate training and test sets or as an oracle for answering membership queries.

In the following paragraphs we describe first the generation of the target context-free grammars, then the experimental setup with learning and test data sets used and finally the results and conclusions that can be drawn.

5.1 Generation of Target Context-free Grammars

To generate the target grammars we follow the process used for the *Omphalos* competition (Starkie et al., 2004). We built 30 different grammars randomly according the following principles. For each grammar, we first fix the number of non-terminals and terminals which are randomly chosen between 4 and 7 for the non-terminals (including the start symbol) and between 2 and 4 for terminal symbols. Then we randomly generate 20 context-free rules in Chomsky normal form such that every non-terminal appears at least once in the left hand side of a grammar rule. In order to avoid the presence of useless rules, we apply two simple procedures: if a non-terminal can not generate any terminal string, a new terminal rule generating one terminal symbol is created for this non-terminal; if a non-terminal can not be reached from the start symbol, we erase it from the grammar (i.e., we remove all the rules containing this non-terminal). From these grammars without useless rules, we force them to generate non finite languages by checking that the start symbol is used at least once in a right hand side of a grammar rule (in average this symbol appears in a right hand side of a rule more than 3 times per grammar).

The main difference with the *Omphalos* generation process is that we do not especially need non-regular languages. Indeed, one of the aim of these experiments is to give an idea on the behavior of the algorithm when its theoretical assumptions are not likely to be valid. From this standpoint, all randomly generated non-finite languages are good candidates as learning targets. However, with a similar principle used for the *Omphalos* competition, we checked that some of the generated grammars can not be easily solved by methods for regular languages. Although we can not decide if these grammars define non regular context-free languages, it ensures us that the target models are at least not too simple.

5.2 Experimental Setup

For each target grammar we generate a learning and a test sample following the *Omphalos* competition requirements. We build the learning sample by first creating a *structurally complete set* of strings for each grammar. This set is built such that for each rule of the target grammar, at least one string of the set can be derived using this rule (Parekh and Honavar, 1996). This would guarantee that the complete learning set would have the minimal amount of information for finding the structure of the grammar. We then complete this learning set by randomly generating new strings from the grammar in order to have a total of 50 examples. We chose arbitrarily this value for two reasons: first it is sufficient to ensure that each sample strictly contains a structurally complete set for each target grammar and secondly we are likely to be far from the guarantees of the identification in the limit framework.

The construction of the test set needs particular attention. Since the learning phase uses a membership oracle, when the hypothesis is being constructed, some new strings may be built and queried for the oracle by picking a substring and a context from the learning sample. Thus, even if the test set does not contain any string of the learning sample, the construction G_0 may consider some strings present in the test set. In order to avoid this drawback, that is, to guarantee that no string of the test could be seen during the construction of the CBFG, each test string has a length of at least 3 times the maximal length of the strings in the learning set, which is by construction the maximal size of the strings queried. According to this procedure, we randomly generate a test set of 1000 strings over the alphabet of terminal symbols used to define the target grammar (1000 examples is twice the size of the small test sets of the *Omphalos* competition). The test sequences are then labeled positive or negative depending on their membership to the language defined by the grammar. We repeat this process until we have the desired number of strings. The ratio between strings in the language and strings outside the language is fixed to be between 40% and 60%.

In order to study the behavior of our algorithm, we define the following setup. For each target context-free grammar, we construct a CBFG by applying the construction $G_0(K,O,F)$ with $K = Sub(S)$ and $F = Con(S)$ where S is a set of strings drawn from the learning set and using the target grammar as the oracle O for the membership queries. We generate different sets S by drawing an increasing number of learning examples (from 2 to 50) from the learning sample of the considered grammar. Then, we evaluate the learned CBFG on the test sample by measuring the accuracy of correct classification. We present the results averaged on the 30 test sets of the different target context-free grammars.

5.3 Results and Discussion

Figure 3 shows the averaged accuracy over the different target grammars according to the number of strings in the learning sample. We can note that a high correct classification rate (nearly 90%) is reached with 20 examples and with only 5 examples an accuracy of 75% is obtained. These results indicate that a relevant hypothesis can be found even with few examples. The standard deviations represented by vertical bars show a good stability of the results from learning sets of 20 strings. This confirms that our algorithm is able to learn partly correct representations even when learning sets may not have a kernel or a fiducial learning set and thus are far from the identification in the limit assumptions.

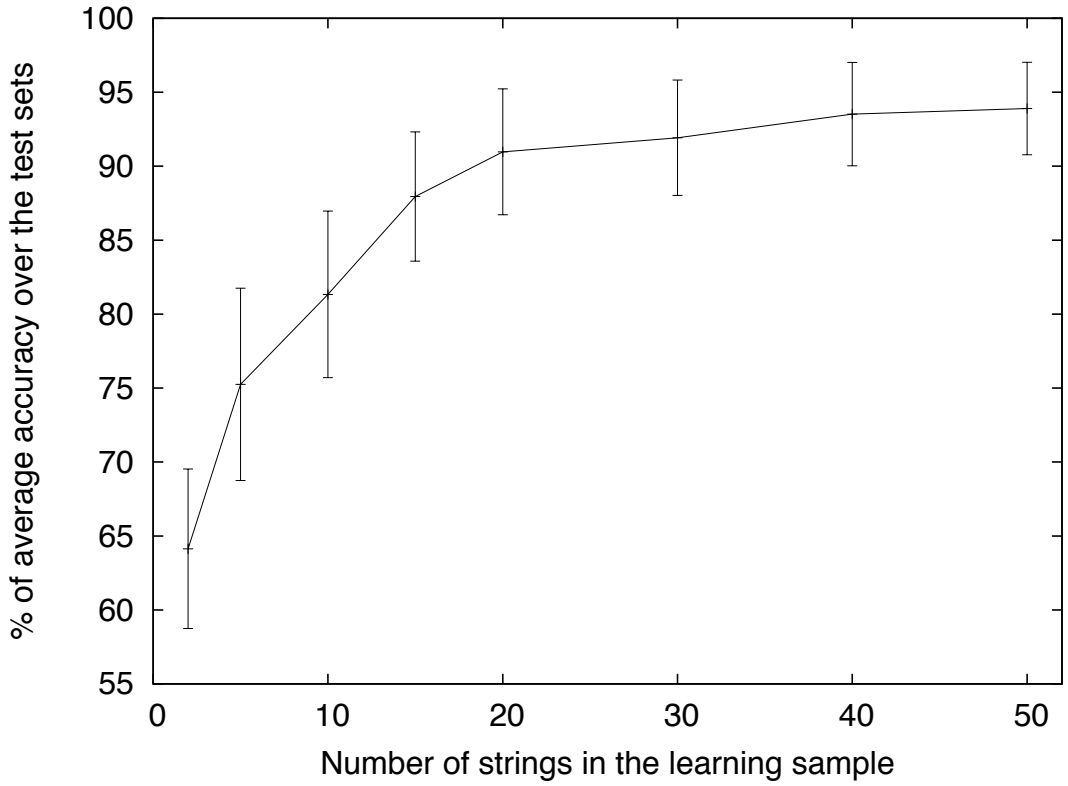


Figure 3: Evolution of the average percentage of correct classification according to the number of learning examples.

The analysis of the behavior of the algorithm in terms of false positive and false negative rates is shown in Table 1. The proportion of false negatives (i.e., positive strings classified as negative) is higher than the proportion of false positives (i.e., negative strings classified as positive), whatever the size of the learning sample is. Thus the output of the algorithm tends more to under-generalise than the converse. As it is generally admitted that over-generalisation is the main trouble when learning from positive examples, this tendency confirms that the algorithm behaves well. However, it is difficult to draw firm conclusions without a natural distribution over negative examples.

Number of strings in S	false positive	false negative
02	07.2 % $\pm$ 12.3	36.4 % $\pm$ 7.0
05	04.5 % $\pm$ 4.7	28.4 % $\pm$ 9.1
10	03.8 % $\pm$ 2.9	22.4 % $\pm$ 8.4
15	03.9 % $\pm$ 2.6	14.1 % $\pm$ 6.6
20	04.1 % $\pm$ 3.0	09.4 % $\pm$ 6.1
30	04.2 % $\pm$ 2.8	07.9 % $\pm$ 5.5
40	03.9 % $\pm$ 2.6	05.6 % $\pm$ 4.7
50	04.4 % $\pm$ 1.6	04.8 % $\pm$ 3.9

Table 1: Average percentage of false positive and false negative rates obtained over the test samples.

The preceding results show that despite its simplicity the algorithm behaved nicely during these experiments, in particular concerning over-generalisation. We focus now on the amount of queries needed by the algorithm for building the CBFG. The growth of the number of requested queries according to the average size of the learning sample is shown in Figure 4 (recall that here the size of the sample means the sum of the string lengths of the sample). While a very worst case analysis of the grammar construction used by G_0 would lead to a complexity in $O(|S|^5)$, we can observe that the number of queries seems to be quadratic, at least in the case of the grammars we consider here. However, the volume of queries used is large, which can be explained by the simplicity of the algorithm. From a practical standpoint, it is clear that much work has to be done in order to try to minimise the number of queries needed by selecting the most informative examples, but this point is out of the scope of the paper.

Finally, we can note that these experiments suffer of the lack of comparison with other approaches. This is due to the fact that, as far as we know, no other algorithm uses a positive learning sample and a membership oracle only. Indeed, since the work of Angluin about the Minimum Adequate Teacher (Angluin, 1988) all algorithms using membership queries are designed with the additional help of equivalence queries. The point of view adopted in this paper is rather theoretical since our aim was to show the relevance of CBFG representations for language learning. However, a perspective of our work is to try to avoid the use of the oracle (by using statistical or simulation methods) which will allow us to compare more easily our approach with other methods.

6. Expressiveness of CBFG

In this section, we compare the expressiveness of CBFG with other well known representations. As noted earlier, we are primarily interested in the class of exact CBFGs—these are CBFGs where the presence of a contextual feature in the representation corresponds exactly to the language theoretic interpretation of the context. The class of unrestricted CBFG is significantly larger, but less relevant.

The algorithm presented in this paper cannot learn the entire class of exact CBFGs, but we conjecture that there are more powerful algorithms that can; see Clark (2009) for some steps in this direction.

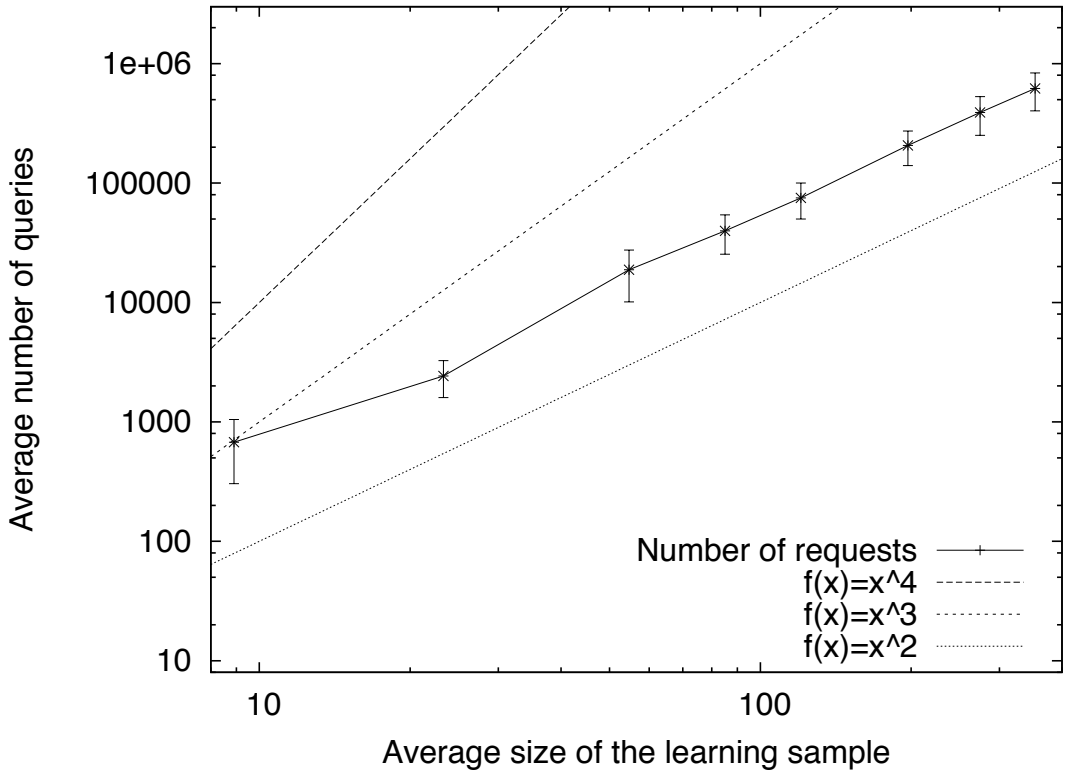


Figure 4: Growth of the number of membership queries versus the average of the total size of the learning sample (using log log scale).

6.1 Exact CBFGs and the Chomsky Hierarchy

We start by examining the class of languages defined by exact CBFGs. We will show that this class

- includes all regular languages
- does not include all context free languages
- includes some non-context-free languages.

This class is thus orthogonal to the Chomsky hierarchy.

6.1.1 REGULAR LANGUAGES

Any regular language can be defined by an exact CBFG. We will show a way of constructing an exact CBFG for any regular language. Suppose we have a regular language L: we consider the left and right residual languages:

$$u^{-1}L = \{w | uw \in L\},$$

$$Lu^{-1} = \{w | wu \in L\}.$$

For any $u \in \Sigma^*$, let $l_{min}(u)$ be the lexicographically shortest element such that $l_{min}^{-1}L = u^{-1}L$. The number of such l_{min} is finite by the Myhill-Nerode theorem, we denote by L_{min} this set, that is, $\{l_{min}(u)|u \in \Sigma^*\}$. We define symmetrically R_{min} for the right residuals ($Lr_{min}^{-1} = Lu^{-1}$).

We define the set of contexts as:

$$F(L) = L_{min} \times R_{min}.$$

$F(L)$ is clearly finite by construction.

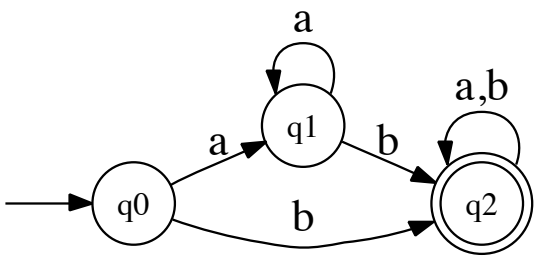


Figure 5: Example of a DFA. The left residuals are defined by $\lambda^{-1}L, a^{-1}L, b^{-1}L$ are the right ones by $L\lambda^{-1}, Lb^{-1}, Lab^{-1}$ (note here that $La^{-1} = L\lambda^{-1}$).

If we consider the regular language defined by the deterministic finite automata of Figure 5, we obtain $L_{min} = \{\lambda, a, b\}$ and $R_{min} = \{\lambda, b, ab\}$ and thus

$$F(L) = \{(\lambda,\lambda),(a,\lambda),(b,\lambda),(\lambda,b),(a,b),(b,b),(\lambda,ab),$$

$$(a,ab),(b,ab)\}.$$

By considering this set of features, we can prove the following lemma:

Lemma 35 *For any strings u, v such that $F_L(u) \supset F_L(v)$ then $C_L(u) \supset C_L(v)$.*

Proof Suppose $F_L(u) \supset F_L(v)$ and let (l,r) be a context in $C_L(v)$. Let l' be the lexicographically shortest element of $\{u : u^{-1}L = l^{-1}L\}$ and r' the lexicographically shortest element of $\{u : Lu^{-1} = Lr^{-1}\}$. By construction we have $(l',r') \in F(L)$ and $l'vr' \in L$, as $vr' \in l'^{-1}L = l^{-1}L$. $F_L(v)$ is contained in $F_L(u)$ therefore we have $l'ur' \in L$. $l'^{-1}L = l^{-1}L$ implies $lur' \in L$. As r' is congruent to r, $lur \in L$. ■

This lemma means that the set of features F is sufficient to represent context inclusion.

Note that the number of congruence classes of a regular language is finite. Each congruence class is represented by a set of contexts $F_L(u)$. Let K_L be finite set of strings formed by taking the lexicographically shortest string from each congruence class. The final grammar can be obtained by combining elements of K_L. For every pair of strings $u, v \in K_L$, we define a rule

$$F_L(uv) \to F_L(u)F_L(v)$$

and we add lexical productions of the form $F_L(a) \to a, a \in \Sigma$.

The following lemma shows the correctness and the exactness of the grammar.

Lemma 36 *For all* $w \in \Sigma^*$, $f_G(w) = F_L(w)$.

Proof *(Sketch)* The proof is in two steps: $f_G(w) \subseteq F_L(w)$ and $F_L(w) \subseteq f_G(w)$. Each step is made by induction on the length of w and uses the rules created to build the grammar, the derivation process of a CBFG and the fiduciality for the second step.

First, we show $\forall w \in \Sigma^*, f_g(w) \subseteq F_L(w)$ by induction on the length of w. For $|w| = 1$, the inclusion is trivial since all the lexical rules $F_L(a) \to a$ are included in the grammar. Suppose that a string w, $|w| = n > 1$, is parsed by the CBFG G, then there exists a cut of w in $uv = w$ and a rule $z \to xy$ in G such that $x \subseteq f_G(u)$ and $y \subseteq f_G(v)$. By induction hypothesis, $x \subseteq F_L(u)$ and $y \subseteq F_L(v)$. By construction of the grammar, there exists two strings $u', v' \in K_L$ such that u, resp. v, belongs to same congruence class than u', resp. v' and the rule $F_L(u'v') \to F_L(u')F_L(v')$ belongs to the productions of the grammar. By induction hypothesis, $x \subseteq F_L(u) = F_L(u')$ and $y \subseteq F_L(v) = F_L(v')$ and thus $f_G(w) \subseteq F_L(w)$.
Second, we prove that $\forall w \in \Sigma^*, F_L(w) \subseteq f_G(w)$ by induction on the length of w. The key point relies on the fact that when a string w is parsed by a CBFG G, there exists a cut of w into $uv = w$ (u, v $\in \Sigma^*$) and a rule $z \to xy$ in G such that $x \subseteq f_G(u)$ and $y \subseteq f_G(v)$. The rule $z \to xy$ is also obtained from a substring from the set used to build the grammar using the F_L function. By the inductive hypothesis we obtain inclusion between f_G and F_L on u and v. ■

For the language of Figure 5, the following set is sufficient to build an exact CBFG: $\{a, b, aa, ab, ba, aab, bb, bba\}$ (this corresponds to all the substrings of *aab* and *bba*). We have:

$$F_L(a) = F(L) \backslash \{(\lambda, \lambda), (a, \lambda)\} \to a,$$

$$F_L(b) = F(L) \to b,$$

$$F_L(aa) = F_L(a) \to F_L(a)F_L(a),$$

$$F_L(ab) = F(L) \to F_L(a)F_L(b) = F_L(a)F(L),$$

$$F_L(ba) = F(L) \to F_L(b)F_L(a) = F(L)F_L(a),$$

$$F_L(bb) = F(L) \to F_L(b)F_L(b) = F(L)F(L),$$

$$F_L(aab) = F_L(bba) = F_L(ab) = F_L(ba).$$

The approach presented here gives a canonical form for representing a regular language by an exact CBFG. Moreover, this is is *complete* in the sense that every context of every substring will be represented by some element of F: this CBFG will completely model the relation between contexts and substrings.

6.1.2 EXACT CBFGS DO NOT INCLUDE ALL CFLS

First, it is clear that the class of exact CBFGs includes some non-regular context-free languages: the grammar defined in Section 3.3 is an exact CBFG for the context-free and non regular language $\{a^n b^n | n > 0\}$, showing the class of exact CBFG has some elements properly in the class of CFGs.

We give now a context-free language L that can not be defined by an exact CBFG:

$$L = \{a^n b | n > 0\} \cup \{a^n c^m | m > n > 0\}.$$

Suppose that there exists an exact CBFG that recognizes it and let N be the length of the biggest feature (i.e., the longest left part of the feature). For any sufficiently large $k > N$, the sequences c^k and c^{k+1} share the same features: $F_L(c^k) = F_L(c^{k+1})$. Since the CBFG is exact we have $F_L(b) \subseteq F_L(c_k)$. Thus any derivation of $a^{k+1}b$ could be a derivation of $a^{k+1}c^k$ which does not belong to the language.

However, this restriction does not mean that the class of exact CBFG is too restrictive for modeling natural languages. Indeed, the example we have given is highly unnatural and such phenomena appear not to occur in attested natural languages.

6.1.3 CBFG AND NON CONTEXT-FREE LANGUAGES

CBFGs are more powerful than CFGs in two respects. First, CBFGs can compactly represent languages like the finite language of all $n!$ permutations of an n-letter alphabet, that have no concise representation as a CFG (Asveld, 2006). Secondly, as we now show, there are some exact CBFGs that are not context-free. In particular, we define a language closely related to the *MIX language* (consisting of strings with an equal number of a's, b's and c's in any order) which is known to be non context-free, and indeed is conjectured to be outside the class of indexed grammars (Boullier, 2003).

Let $M = \{\{a,b,c\}^+\}$, the set of all strings of length at least one that can be built on the alphabet $\{a,b,c\}$. We consider now the language

$$L = L_{abc} \cup L_{ab} \cup L_{ac} \cup \{a'a, b'b, c'c, dd', ee', ff'\} :$$

$$L_{ab} = \{wd | w \in M, |w|_a = |w|_b\},$$

$$L_{ac} = \{we | w \in M, |w|_a = |w|_c\},$$

$$L_{abc} = \{wf | w \in M, |w|_a = |w|_b = |w|_c\}.$$

In order to define a CBFG recognizing L, we have to select features (contexts) that can represent exactly the intrinsic components of the languages composing L. We propose to use the following set of features for each sublanguage:

- For L_{ab}: (λ, d) and $(\lambda, ad), (\lambda, bd)$.
- For L_{ac}: (λ, e) and $(\lambda, ae), (\lambda, ce)$.
- For L_{abc}: (λ, f').
- For the letters a', b', c', a, b, c we add: $(\lambda, a), (\lambda, b), (\lambda, c), (a', \lambda), (b', \lambda), (c', \lambda)$.
- For the letters d, e, f, d', e', f' we add; $(\lambda, d'), (\lambda, e'), (\lambda, f'), (d, \lambda), (e, \lambda), (f, \lambda)$.

Here, L_{ab} will be represented by (λ, d), but we will use $(\lambda, ad), (\lambda, bd)$ to define the internal derivations of elements of L_{ab}. The same idea holds for L_{ac} with (λ, e) and $(\lambda, ae), (\lambda, ce)$.

For the lexical rules and in order to have an exact CBFG, note the special case for a, b, c:

$$\{(\lambda, bd), (\lambda, ce), (a', \lambda)\} \to a,$$

$$\{(\lambda, ad), (b', \lambda)\} \to b,$$

$$\{(\lambda, ae), (c', \lambda)\} \to c.$$

For the nine other letters, each one is defined with only one context, for example using the rule $\{(\lambda, d')\} \to d$.

For the production rules, the most important one is: $(\lambda, \lambda) \to \{(\lambda, d), (\lambda, e)\}, \{(\lambda, f')\}$.

Indeed, this rule, with the presence of two contexts in one of categories, means that an element of the language has to be derived so that it has a prefix u such that $f_G(u) \supseteq \{(\lambda, d), (\lambda, e)\}$. This means u is both an element of L_{ab} and L_{ac}. This rule represents the language L_{abc} since $\{(\lambda, f')\}$ can only represent the letter f.

The other parts of the language will be defined by the following rules:

$$(\lambda, \lambda) \to \{(\lambda, d)\}, \{(\lambda, d')\},$$

$$(\lambda, \lambda) \to \{(\lambda, e)\}, \{(\lambda, e')\},$$

$$(\lambda, \lambda) \to \{(\lambda, a)\}, \{(\lambda, bd), (\lambda, ce), (a', \lambda)\},$$

$$(\lambda, \lambda) \to \{(\lambda, b)\}, \{(\lambda, ad), (b', \lambda)\},$$

$$(\lambda, \lambda) \to \{(\lambda, c)\}, \{(\lambda, ae), (c', \lambda)\},$$

$$(\lambda, \lambda) \to \{(\lambda, d')\}, \{(d, \lambda)\},$$

$$(\lambda, \lambda) \to \{(\lambda, e')\}, \{(e, \lambda)\},$$

$$(\lambda, \lambda) \to \{(\lambda, f')\}, \{(f, \lambda)\}.$$

This set of rules is incomplete, since for representing L_{ab}, the grammar must contain the rules ensuring to have the same number of a's and b's, and similarly for L_{ac}. To lighten the presentation here, the complete grammar is presented in Appendix.

We claim this is an exact CBFG for a context-sensitive language. L is not context-free since if we intersect L with the regular language $\Sigma^* d$, we get an instance of the non context-free MIX language (with d appended). The exactness comes from the fact that we chose the contexts in order to ensure that strings belonging to a sublanguage can not belong to another one and that the derivation of a substring will provide all the possible correct features with the help of the union of all the possible derivations.

Note that the MIX language on its own is not definable by an exact CBFG: it is only when other parts of the language can distributionally define the appropriate partial structures that we can get context sensitive languages. Far from being a limitation of this formalism (a bug), we argue this is a feature: it is only in rather exceptional circumstances that we will get properly context sensitive languages. This formalism thus potentially accounts not just for the existence of non context-free natural languages but also for their rarity.

6.2 Inexact CBFGs

We are less interested in the class of all CBFGs: these are CBFGs where the contexts are just arbitrary features and there is no relation between $f_G(u)$ and $C_L(u)$ except for the presence of (λ, λ). However, it is important to understand the language theoretic power of this class as this upper bounds the hypothesis class of the algorithm, and is easier to analyse.

6.2.1 CONTEXT-FREE GRAMMARS

First, we note that this class contains all context-free languages. Given a context-free language, that does not include the empty string, we can take a CFG in Chomsky normal form and convert it directly into a CBFG. Let V be the set of non-terminals of such a CFG. We pick an arbitrary set of distinct contexts to represent the elements of V, subject only to the constraint that S corresponds to (λ,λ). Let $C(N)$ be the context corresponding to the non-terminal N. For every production rule in the CFG of the form $N \to PQ$, we add a CBFG production $\{C(N)\} \to \{C(P)\},\{C(Q)\}$. For every production in the CFG of the form $N \to a$, we add a CBFG production to P_L of the form $\{C(N)\} \to a$. It is easy to see that this will define the same language.

6.2.2 RANGE CONCATENATION GRAMMARS

While CBFG formalism has some relationship to a context-free grammar, and some to a semi-Thue system (also known as a string rewriting system), it is not formally identical to either of these. One exact equivalence is to a restricted subset of Range Concatenation Grammars; a very powerful formalism (Boullier, 2000). We include the following relationship, but suggest that the reader unfamiliar with RCGs proceeds to the discussion of the relationship with the more familiar class of context-free grammars.

Lemma 37 *For every CBFG G, there is a non-erasing positive range concatenation grammar of arity one, in 2-var form that defines the same language.*

Proof Suppose $G = \langle F,P,P_L,\Sigma\rangle$. Define a RCG with a set of predicates equal to F and the following clauses, and the two variables U,V. For each production $x \to yz$ in P, for each $f \in x$, where $y = \{g_1,\ldots g_i\}, z = \{h_1,\ldots h_j\}$ add clauses

$$f(UV) \to g_1(U),\ldots g_i(U),h_1(V),\ldots h_j(V).$$

For each lexical production $\{f_1 \ldots f_k\} \to a$ add clauses

$$f_i(a) \to \varepsilon.$$

It is straightforward to verify that $f(w) \vdash \varepsilon$ iff $f \in f_G(w)$. ■

6.2.3 CONJUNCTIVE GRAMMAR

A tighter correspondence is to the class of Conjunctive Grammars (Okhotin, 2001), invented independently of RCGs.

Definition 38 *A conjunctive grammar is defined as a quadruple $\langle\Sigma,N,P,S\rangle$, in which: Σ is the alphabet; N is the set of non terminal symbols; P is the set of rules, each of the form $A \to \alpha_1\&\ldots\&\alpha_m$, where $A \in V$ and $\forall i < m$, $\alpha_i \in (V \cup \Sigma)^*$; $S \in N$ is the start symbol.*

In this formalism, a string w is derived from $A \in V$ iff there exists a rule $A \to \alpha_1\&\ldots\&\alpha_m$ in P and for all $i < m$, α_i derives w.

We claim that for every every language L generated by a conjunctive grammar there is a CBFG representing $L\#$ (where the special character # is not included in the original alphabet).

Suppose we have a conjunctive grammar $G = \langle \Sigma, N, P, S \rangle$ in binary normal form (as defined in Okhotin, 2003). We construct the equivalent CBFG $G' = \langle F, P', P_L, \Sigma \rangle$ as followed:

- For every letter a we add a context (l_a, r_a) to F such that $l_a a r_a \in L$;
- For every rules $X \to a$ in P, we create a rule $\{(l_a, r_a)\} \to a$ in P_L.
- For every non terminal $X \in N$, for every rule $X \to P_1 Q_1 \& \ldots \& P_n Q_n$ we add distinct contexts $\{(l_{P_i Q_i}, r_{P_i Q_i})\}$ to F, such that for all i it exists u_i, $l_{P_i Q_i} u_i r_{P_i Q_i} \in L$ and $P_i Q_i \stackrel{*}{\Rightarrow}_G u_i$;
- Let $F_{X,j} = \{(l_{P_i Q_i}, r_{P_i Q_i}) : \forall i\}$ the set of contexts corresponding to the j^{th} rule applicable to X. For all $(l_{P_i Q_i}, r_{P_i Q_i}) \in F_{X,j}$, we add to P' the rules $(l_{P_i Q_i}, r_{P_i Q_i}) \to F_{P_i,k} F_{Q_i,l}$ $(\forall k, l)$.
- We add a new context (w, λ) to F such that $S \stackrel{*}{\Rightarrow}_G w$ and $(w, \lambda) \to \#$ to P_L;
- For all j, we add to P' the rule $(\lambda, \lambda) \to F_{S,j}\{(w, \lambda)\}$.

It can be shown that this construction gives an equivalent CBFG.

7. Discussion and Conclusion

One of the main objective of our approach is to provide a framework that helps to bridge the gap between theoretical methods of grammatical inference and the structured representations required in linguistics. We provide a conclusion and a discussion of our work according to these two standpoints.

7.1 Grammatical Inference

In this paper, we have presented a new formalism the *Contextual Binary Feature Grammars* and shown its relevance for representing a large class of languages. We have proposed a learning algorithm using only membership queries and shown that this algorithm can identify in the limit the class of context-free languages satisfying the FCP and FKP assumptions. First of all, we should establish how large the class of languages with the FCP and the FKP is: it includes all finite languages and all regular languages, since the set of congruence classes is finite for finite state languages. It similarly includes the context-free substitutable languages (Clark and Eyraud, 2007), since every string in a substitutable language belongs to only one syntactic congruence class. As already stated it does not include all CFLs since not all CFLs have the FCP and/or the FKP. However it does include languages like the Dyck languages of arbitrary order, Lukacevic language and most other classic simple examples. As a special case consider the equivalence relation between contexts $f \cong_L f'$ iff $\forall u$ we have that $f \odot u \in L$ iff $f' \odot u \in L$. The class of CFLs where the context distribution of every string is a finite union of equivalence classes of contexts clearly has both the FKP and the FCP.

If we now focus on the algorithm proposed: it is relatively simple but has two main drawbacks. First, the algorithm is not conservative since once we have found the correct language, the representation may change—if the feature set found is not fiducial—until the fiduciality is reached. Second, the CBFG output by the algorithm may not be consistent with some answers provided by the oracle. Indeed, when the algorithm checks the fiduciality of the feature set F, the membership of new strings is tested. These strings do not appear in the list of learning examples given to the oracle but are built from all the possible contexts and substrings that can be extracted from this list. Then, it is

possible that, among these new strings, some of them belong to the target language but are not recognized by the current grammar. In this case, the output grammar is nevertheless not modified. We can imagine a procedure that changes the grammar by adding these new positive strings for building the CBFG, however this could lead to having to deal with an exponential number of strings. Thus, a more reasonable procedure is to wait for these strings in the positive data presentation. One proposal for future work, from these two remarks, is a new learning algorithm that overcomes these drawbacks.

One important point is whether this result can be extended to a result which also bounds the number of samples as a polynomial function of the size of the representation. A preliminary result in this direction is presented in Clark (2010), which presents a polynomial result using the Minimally Adequate Teacher model of Angluin (1987). It seems likely that it will be possible to extend that result, which uses only context-free grammars, to the class of CBFGs.

Our approach to context-free grammatical inference is based on a generalisation of distributional learning, following the work of Clark and Eyraud (2007). The current state of the art in context-free inductive inference from flat unstructured examples only has been rather limited. When learning from stochastic data or using a membership oracle, it is possible to have powerful results, if we allow exponential computation (see for example Horning, 1969). The main contribution of this paper is to show that efficient learning is possible, with an appropriate representation. We currently rely on using a membership oracle, but under suitable assumptions about distributions, it should be possible to get a PAC-learning result for this class along the lines of Clark (2006), placing some bounds on the number of features required. Another interesting and important issue is the adaptation of this approach to stochastic languages.

We have focused on context-free grammatical inference, however, we have shown that our representation is also relevant for modeling non context-free languages. Then, another perspective of this work is to study learnability results for larger classes of languages. This would allow us to compare with other formalisms such as External Contextual Grammars (Boullier, 2001; Mitrana, 2005) and other learning methods dealing with non context-free languages (Oates et al., 2006; Yoshinaka, 2009).

7.2 Linguistics

The field of grammatical inference has close relations to the study of language acquisition. Attempts to model natural languages with context-free grammars require additional machinery: natural language categories such as noun phrases contain many overlapping subclasses with features such as case, number, gender and similarly for verbal categories. Modelling this requires either an exponential explosion of the number of non-terminals employed or a switch to a richer set of features. Our formalism can be seen as a first step to integrate such features.

While we have implemented the algorithm described here, and verified that it works in accordance with theory on small artificial examples, there are a number of modifications that would need to be made before it can be applied to real grammar induction on natural language. First, the algorithm is very naive; in practice a more refined algorithm could select both the kernel and the feature set in a more sophisticated way. Secondly, considering features that correspond to individual contexts may be too narrow a definition for natural language given the well known problems of data sparseness and it will be necessary to switch to features corresponding to sets of contexts, which may overlap. Thus for example one might have features that correspond to sets of contexts of the

form $F(u,v) = \{(lu, vr) | l, r \in \Sigma^*\}$. This would take this approach closer to methods that have been shown to be effective in unsupervised learning in NLP (Klein and Manning, 2004) where typically $|u| = |v| = 1$. In any event, we think such modifications will be necessary for the acquisition of non context-free languages. Finally, at the moment the algorithm has polynomial update time, but in the worst case, there are deterministic finite state automata such that the size of the smallest kernel will be exponential in the number of states. There are, however, natural algorithms for generalising the productions by removing features from the right hand sides of the rules; this would have the effect of accelerating the convergence of the algorithm, and removing or weakening the requirement for the Finite Kernel Property.

Acknowledgments

This work was supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.

We would like to thank Ryo Yoshinaka for very helpful discussions, and the anonymous reviewers for suggestions and comments that have greatly improved the paper. We also want to thank Jean-Marie Madiot who worked on the first version of the experiments presented in this paper.

Appendix A.

We give here an explicit exact CBFG for the following non context-free language

$$L = L_{abc} \cup L_{ab} \cup L_{ac} \cup \{a'a, b'b, c'c, dd', ee', ff'\}$$

defined on the alphabet $\Sigma = \{a, b, c, d, e, f, a', b', c', d', e', f'\}$ and such that:

$$L_{ab} = \{wd | w \in \{a,b,c\}^+, |w|_a = |w|_b\},$$

$$L_{ac} = \{we | w \in \{a,b,c\}^+, |w|_a = |w|_c\},$$
$$L_{abc} = \{wf | w \in \{a,b,c\}^+, |w|_a = |w|_b = |w|_c\}.$$

Here is the list of productions of the grammar.

$$\{(\lambda,\lambda)\} \to \{(\lambda,d),(\lambda,e)\},\{(\lambda,f')\},$$
$$\{(\lambda,\lambda)\} \to \{(\lambda,d)\},\{(\lambda,d')\},$$
$$\{(\lambda,\lambda)\} \to \{(\lambda,e)\},\{(\lambda,e')\},$$
$$\{(\lambda,\lambda)\} \to \{(\lambda,a)\},\{(\lambda,bd),(\lambda,ce),(a',\lambda)\},$$
$$\{(\lambda,\lambda)\} \to \{(\lambda,b)\},\{(\lambda,ad),(b',\lambda)\},$$
$$\{(\lambda,\lambda)\} \to \{(\lambda,c)\},\{(\lambda,ae),(c',\lambda)\},$$
$$\{(\lambda,\lambda)\} \to \{(\lambda,d')\},\{(d,\lambda)\},$$

$$\{(\lambda,\lambda)\} \rightarrow \{(\lambda,e')\},\{(e,\lambda)\},$$
$$\{(\lambda,\lambda)\} \rightarrow \{(\lambda,f')\},\{(f,\lambda)\},$$

$$\{(\lambda,d)\} \rightarrow \{(\lambda,d)\},\{(\lambda,d)\},$$
$$\{(\lambda,d)\} \rightarrow \{(\lambda,ad)\},\{(\lambda,bd)\},$$
$$\{(\lambda,d)\} \rightarrow \{(\lambda,bd)\},\{(\lambda,ad)\},$$
$$\{(\lambda,d)\} \rightarrow \{(\lambda,d)\},\{(\lambda,ad),(\lambda,ae),(c',\lambda)\},$$
$$\{(\lambda,d)\} \rightarrow \{(\lambda,ad),(\lambda,ae),(c',\lambda)\},\{(\lambda,d)\},$$

$$\{(\lambda,ad)\} \rightarrow \{(\lambda,ad),(\lambda,ae),(c',\lambda)\},\{(\lambda,ad)\},$$
$$\{(\lambda,ad)\} \rightarrow \{(\lambda,ad)\},\{(\lambda,ad),(\lambda,ae),(c',\lambda)\},$$
$$\{(\lambda,ad)\} \rightarrow \{(\lambda,ad),(b',\lambda)\},\{(\lambda,d)\},$$
$$\{(\lambda,ad)\} \rightarrow \{(\lambda,d)\},\{(\lambda,ad),(b',\lambda)\},$$

$$\{(\lambda,bd)\} \rightarrow \{(\lambda,ad),(\lambda,ae),(c',\lambda)\},\{(\lambda,bd)\},$$
$$\{(\lambda,bd)\} \rightarrow \{(\lambda,bd)\},\{(\lambda,ad),(\lambda,ae),(c',\lambda)\},$$
$$\{(\lambda,bd)\} \rightarrow \{(\lambda,bd),(\lambda,ce),(a',\lambda)\},\{(\lambda,d)\},$$
$$\{(\lambda,bd)\} \rightarrow \{(\lambda,d)\},\{(\lambda,bd),(\lambda,ce),(a',\lambda)\},$$

$$\{(\lambda,e)\} \rightarrow \{(\lambda,e)\},\{(\lambda,e)\},$$
$$\{(\lambda,e)\} \rightarrow \{(\lambda,ae)\},\{(\lambda,ce)\},$$
$$\{(\lambda,e)\} \rightarrow \{(\lambda,ce)\},\{(\lambda,ae)\},$$
$$\{(\lambda,e)\} \rightarrow \{(\lambda,e)\},\{(\lambda,ad),(b',\lambda)\},$$
$$\{(\lambda,e)\} \rightarrow \{(\lambda,ad),(b',\lambda)\},\{(\lambda,e)\},$$

$$\{(\lambda,ae)\} \rightarrow \{(\lambda,ad),(b',\lambda)\},\{(\lambda,ae)\},$$
$$\{(\lambda,ae)\} \rightarrow \{(\lambda,ae)\},\{(\lambda,ad),(b',\lambda)\},$$
$$\{(\lambda,ae)\} \rightarrow \{(\lambda,ad),(\lambda,ae),(c',\lambda)\},\{(\lambda,e)\},$$
$$\{(\lambda,ae)\} \rightarrow \{(\lambda,e)\},\{(\lambda,ad),(\lambda,ae),(c',\lambda)\},$$

$$\{(\lambda,ce)\} \rightarrow \{(\lambda,ad),(b',\lambda)\},\{(\lambda,ce)\},$$
$$\{(\lambda,ce)\} \rightarrow \{(\lambda,ce)\},\{(\lambda,ad),(b',\lambda)\},$$
$$\{(\lambda,ce)\} \rightarrow \{(\lambda,bd),(\lambda,ce),(a',\lambda)\},\{(\lambda,e)\},$$

$$\{(\lambda, ce)\} \to \{(\lambda, e)\}, \{(\lambda, bd), (\lambda, ce), (a', \lambda)\},$$

$$\{(\lambda, bd), (\lambda, ce), (a', \lambda)\} \to a,$$
$$\{(\lambda, ad), (b', \lambda)\} \to b,$$
$$\{(\lambda, ae), (c', \lambda)\} \to c,$$
$$\{(\lambda, d')\} \to d,$$
$$\{(\lambda, e')\} \to e,$$
$$\{(\lambda, f')\} \to f,$$

$$\{(\lambda, a)\} \to a',$$
$$\{(\lambda, b)\} \to b',$$
$$\{(\lambda, c)\} \to c',$$
$$\{(d, \lambda)\} \to d',$$
$$\{(e, \lambda)\} \to e',$$
$$\{(f, \lambda)\} \to f'.$$

References

P. Adriaans. Learning shallow context-free languages under simple distributions. *Algebras, Diagrams and Decisions in Language, Logic and Computation*, 127, 2002.

D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

D. Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1988. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1022821128753.

P.R.J. Asveld. Generating all permutations by context-free grammars in Chomsky normal form. *Theoretical Computer Science*, 354(1):118–130, 2006.

L. Boasson and G. Senizergues. NTS languages are deterministic and congruential. *J. Comput. Syst. Sci.*, 31(3):332–342, 1985. ISSN 0022-0000. doi: http://dx.doi.org/10.1016/0022-0000(85)90056-X.

P. Boullier. A Cubic Time Extension of Context-Free Grammars. *Grammars*, 3:111–131, 2000.

P. Boullier. From contextual grammars to range concatenation grammars. *Electronic Notes on Theoretical Computer Science*, 53, 2001.

P. Boullier. Counting with range concatenation grammars. *Theoretical Computer Science*, 293(2): 391–416, 2003.

R.C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In R.C. Carrasco and J. Oncina, editors, *Proceedings ICGI'94*, volume 862 of *LNAI*, pages 139–150. Springer, 1994.

N. Chomsky. *Knowledge of Language : Its Nature, Origin, and Use*. Praeger, 1986.

A. Clark. PAC-learning unambiguous NTS languages. In *Proceedings of the 8th International Colloquium on Grammatical Inference (ICGI)*, pages 59–71, 2006.

A. Clark. A learnable representation for syntax using residuated lattices. In *Proceedings of the conference on Formal Grammar*, Bordeaux, France, 2009.

A. Clark. Distributional learning of some context-free languages with a minimally adequate teacher. In José M. Sempere and Pedro García, editors, *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI*, pages 24–37. Springer, 2010.

A. Clark and R. Eyraud. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745, 2007.

C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27 (2):125–138, 1997. ISSN 0885-6125.

F. Denis, A. Lemay, and A. Terlutte. Learning regular languages using rfsas. *Theor. Comput. Sci.*, 313(2):267–294, 2004. ISSN 0304-3975.

R. Eyraud, C. de la Higuera, and J.C. Janodet. Lars: A learning algorithm for rewriting systems. *Machine Learning*, 66(1):7–31, 2007.

G. Gazdar, E. Klein, G. Pullum, and I. Sag. *Generalised Phrase Structure Grammar*. Basil Blackwell, 1985.

E.M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

Z. Harris. Distributional structure. *Word*, 10(2-3):146–62, 1954.

C. De La Higuera and J. Oncina. Inferring deterministic linear languages. In *COLT '02: 15th Annual Conference on Computational Learning Theory*, pages 185–200. Springer-Verlag, 2002.

J.J. Horning. *A Study of Grammatical Inference*. PhD thesis, Stanford University, Computer Science Department, California, 1969.

A.K. Joshi and Y. Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, Berlin, New York, 1997.

D. Klein and C.D. Manning. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 478–485, 2004.

K.J. Lang, B.A. Pearlmutter, and R. Price. Results of the abbadingo one dfa learning competition and a new evidence driven state merging algorithm. In *Grammatical Inference: Algorithms and Applications; 4th International Colloquium on Grammatical Inference*, pages 1–12. Springer-Verlag, 1998. URL `http://abbadingo.cs.unm.edu/`.

P. Langley and S. Stromsten. Learning context-free grammars with a simplicity bias. In *Proceedings of the Eleventh European Conference on Machine Learning*, pages 220–228. Springer-Verlag, 2000.

S. Marcus. *Algebraic Linguistics; Analytical Models*. Academic Press, N. Y., 1967.

V. Mitrana. Marcus external contextual grammars: From one to many dimensions. *Fundamenta Informaticae*, 54:307–316, 2005.

K. Nakamura and M. Matsumoto. Incremental learning of context-free grammars based on bottom-up parsing and search. *Pattern Recognition*, 38(9):1384–1392, 2005.

T. Oates, T. Armstrong, L. Becerra-Bonache, and M. Atamas. Inferring grammars for mildly context sensitive languages in polynomial-time. In *Proceedings of the International Colloquium on Grammatical Inference (ICGI) 2006*, volume 4201 of *LNCS*, pages 137–147. Springer, 2006.

A. Okhotin. Conjunctive grammars. *J. Autom. Lang. Comb.*, 6(4):519–535, 2001. ISSN 1430-189X.

A. Okhotin. An overview of conjunctive grammars. *Formal Language Theory Column, bulletin of the EATCS*, 79:145–163, 2003.

R. Parekh and V. Honavar. An incremental interactive algorithm for regular grammar inference. In *Grammatical Inference : Learning Syntax from Sentences; 3rd International Colloquium on Grammatical Inference*, pages 238–250. Springer-Verlag, 1996.

L. Pitt. Inductive inference, DFA's, and computational complexity. In *Proceedings of the International Workshop on Analogical and Inductive Inference*, pages 18–44. Springer-Verlag, 1989.

B. Starkie, F. Coste, and M. Van Zaanen. The omphalos context-free grammar learning competition. In *Grammatical Inference: Algorithms and Applications; 7th International Colloquium on Grammatical Inference*, pages 16–27. Springer, 2004. URL `http://www.irisa.fr/Omphalos/`.

B. Starkie, M. van Zaanen, and D. Estival. The Tenjinno machine translation competition. In *Grammatical Inference: Algorithms and Applications, 8th International Colloquium on Grammatical Inference*, volume 4201 of *Lecture Notes in Computer Science*, pages 214–226. Springer-Verlag, 2006.

T. Yokomori. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 298(1):179–206, 2003.

R. Yoshinaka. Identification in the limit of k,l-substitutable context-free languages. In *ICGI '08: Proceedings of the 9th International Colloquium on Grammatical Inference*, pages 266–279, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88008-0.

R. Yoshinaka. Learning mildly context-sensitive languages with multidimensional substitutability from positive data. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory (ALT)*, volume 5809 of *LNCS*, pages 278–292. Springer, 2009.

Journal of Machine Learning Research 11 (2010) 2745-2783 Submitted 5/10; Revised 9/10; Published 10/10

Mean Field Variational Approximation for Continuous-Time Bayesian Networks*

Ido Cohn† IDO_COHN@CS.HUJI.AC.IL
Tal El-Hay† TALE@CS.HUJI.AC.IL
Nir Friedman NIR@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University
Jerusalem 91904, Israel

Raz Kupferman RAZ@MATH.HUJI.AC.IL
Institute of Mathematics
The Hebrew University
Jerusalem 91904, Israel

Editor: Manfred Opper

Abstract

Continuous-time Bayesian networks is a natural structured representation language for multi-component stochastic processes that evolve continuously over time. Despite the compact representation provided by this language, inference in such models is intractable even in relatively simple structured networks. We introduce a mean field variational approximation in which we use a product of *inhomogeneous* Markov processes to approximate a joint distribution over trajectories. This variational approach leads to a globally consistent distribution, which can be efficiently queried. Additionally, it provides a lower bound on the probability of observations, thus making it attractive for learning tasks. Here we describe the theoretical foundations for the approximation, an efficient implementation that exploits the wide range of highly optimized ordinary differential equations (ODE) solvers, experimentally explore characterizations of processes for which this approximation is suitable, and show applications to a large-scale real-world inference problem.

Keywords: continuous time Markov processes, continuous time Bayesian networks, variational approximations, mean field approximation

1. Introduction

Many real-life processes can be naturally thought of as evolving continuously in time. Examples cover a diverse range, starting with classical and modern physics, but also including robotics (Ng et al., 2005), computer networks (Simma et al., 2008), social networks (Fan and Shelton, 2009), gene expression (Lipshtat et al., 2005), biological evolution (El-Hay et al., 2006), and ecological systems (Opper and Sanguinetti, 2007). A joint characteristic of all above examples is that they are complex systems composed of multiple components (e.g., many servers in a server farm and multiple residues in a protein sequence). To realistically model such processes and use them in

*. A preliminary version of this paper appeared in the Proceedings of the Twenty Fifth Conference on Uncertainty in Artificial Intelligence, 2009 (UAI 09).
†. These authors contributed equally.

making sensible predictions we need to learn how to reason about systems that are composed of multiple components and evolve continuously in time.

Generally, when an evolving system is modeled with sufficient detail, its evolution in time is Markovian; meaning that its future state it determined by its present state—whether in a deterministic or random sense—independently of its past states. A traditional approach to modeling a multi-component Markovian process is to discretize the entire time interval into regular time slices of fixed length and represent its evolution using a *Dynamic Bayesian network*, which compactly represents probabilistic transitions between consecutive time slices (Dean and Kanazawa, 1989; Murphy, 2002; Koller and Friedman, 2009). However, as thoroughly explained in Nodelman et al. (2003), discretization of a time interval often leads either to modeling inaccuracies or to an unnecessary computational overhead. Therefore, in recent years there is a growing interest in modeling and reasoning about multi-component stochastic processes in continuous time (Nodelman et al., 2002; Ng et al., 2005; Rajaram et al., 2005; Gopalratnam et al., 2005; Opper and Sanguinetti, 2007; Archambeau et al., 2007; Simma et al., 2008).

In this paper we focus on *continuous-time Markov processes* having a discrete product state space $S = S_1 \times S_2 \times \cdots \times S_D$, where D is the number of components and the size of each S_i is finite. The dynamics of such processes that are also *time-homogeneous* can be determined by a single rate matrix whose entries encode transition rates among states. However, as the size of the state space is exponential in the number of components so does the size of the transition matrix. *Continuous-time Bayesian networks* (CTBNs) provide an elegant and compact representation language for multi-component processes that have a sparse pattern of interactions (Nodelman et al., 2002). Such patterns are encoded in CTBNs using a directed graph whose nodes represent components and edges represent direct influences among them. The instantaneous dynamics of each component depends only on the state of its parents in the graph, allowing a representation whose size scales linearly with the number of components and exponentially only with the indegree of the nodes of the graph.

Inference in multi-component temporal models is a notoriously hard problem (Koller and Friedman, 2009). Similar to the situation in discrete time processes, inference in CTBNs is exponential in the number of components, even with sparse interactions (Nodelman et al., 2002). Thus, we have to resort to approximate inference methods. The recent literature has adapted several strategies from discrete graphical models to CTBNs in a manner that attempts to exploit the continuous-time representation, thereby avoiding the drawbacks of discretizing the model.

One class of approximations includes sampling-based approaches, where Fan and Shelton (2008) introduce a likelihood-weighted sampling scheme, and more recently El-Hay et al. (2008) introduce a Gibbs-sampling procedure. The complexity of the Gibbs sampling procedure has been shown to naturally adapt to the rate of each individual component. Additionally it yields more accurate answers with the investment of additional computation. However, it is hard to bound the required time in advance, tune the stopping criteria, or estimate the error of the approximation.

An alternative class of approximations is based on *variational principles*. Recently, Nodelman et al. (2005b) and Saria et al. (2007) introduced an *Expectation Propagation* approach, which can be roughly described as a local message passing scheme, where each message describes the dynamics of a single component over an interval. This message passing procedure can be efficient. Moreover it can automatically refine the number of intervals according to the complexity of the underlying system. Nonetheless, it does suffer from several caveats. On the formal level, the approximation has no convergence guarantees. Second, upon convergence, the computed marginals do not neces-

sarily form a globally consistent distribution. Third, it is restricted to approximations in the form of piecewise-homogeneous messages on each interval. Thus, the refinement of the number of intervals depends on the fit of such homogeneous approximations to the target process. Finally, the approximation of Nodelman *et al* does not provide a provable approximation on the likelihood of the observation—a crucial component in learning procedures.

Here, we develop an alternative variational approximation, which provides a different tradeoff. We use the strategy of structured variational approximations in graphical models (Jordan et al., 1999), and specifically the variational approach of Opper and Sanguinetti (2007) for approximate inference in latent Markov Jump Processes, a related class of models (see below for more elaborate comparison). The resulting procedure approximates the posterior distribution of the CTBN as a product of independent components, each of which is an inhomogeneous continuous-time Markov process. We introduce a novel representation that is both natural and allows numerically stable computations. By using this representation, we derive an iterative variational procedure that employs passing information between neighboring components as well as solving a small set of differential equations (ODEs) in each iteration. The latter allows us to employ highly optimized standard ODE solvers in the implementation. Such solvers use an adaptive step size, which as we show is more efficient than any fixed time interval approximation.

We finally describe how to extend the proposed procedure to branching processes and particularly to models of molecular evolution, which describe historical dynamics of biological sequences that employ many interacting components. Our experiments on this domain demonstrate that our procedure provides a good approximation both for the likelihood of the evidence and for the expected sufficient statistics. In particular, the approximation provides a lower-bound on the likelihood, and thus is attractive for use in learning.

The paper is organized as follows: In Section 2 we review continuous-time models and inference problems in such models. Section 3 introduces a general variational principle for inference using a novel parameterization. In Section 4 we apply this principle to a family of factored representations and show how to find an optimal approximation within this family. Section 5 discusses related work. Section 6 gives an initial evaluation. Section 7 presents branching process and further experiments, and Section 8 discusses our results.

2. Foundations

CTBNs are based on the framework of *continuous-time Markov processes (CTMPs)*. In this section we begin by briefly describing CTMPs. See, for example, Gardiner (2004) and Chung (1960) for a thorough introduction. Next we review the semantics of CTBNs. We then discuss inference problems in CTBNs and the challenges they pose.

2.1 Continuous Time Markov Processes

A *continuous-time stochastic process with state space S* is an uncountable collection of S-valued random variables $\{X^{(t)} : t \geq 0\}$ where $X^{(t)}$ describes the state of the system at time t. Systems with multiple components are described by state spaces that are Cartesian products of spaces, S_i, each representing the state of a single component. In this paper we consider a D-component stochastic process $X^{(t)} = (X_1^{(t)}, \ldots, X_D^{(t)})$ with state space $S = S_1 \times S_2 \times \ldots \times S_D$, where each S_i is finite. The states in S are denoted by vectors, $x = (x_1, \ldots, x_D)$.

A *continuous-time Markov process* is a continuous-time stochastic process in which the joint distribution of every finite subset of random variables $X^{(t_0)}, X^{(t_1)}, \ldots, X^{(t_K)}$, where $t_0 < t_1 < \cdots < t_K$, satisfies the conditional independence property, also known as the Markov property:

$$\Pr(X^{(t_K)} = x_K | X^{(t_{K-1})} = x_{K-1}, \ldots, X^{(t_0)} = x_0) = \Pr(X^{(t_K)} = x_K | X^{(t_{K-1})} = x_{K-1}).$$

In simple terms, the knowledge of the state of the system at a certain time make its states at later times independent of its states at former times. In that case the distribution of the process is fully determined by the conditional probabilities of random variable pairs $\Pr(X^{(t+s)} = y | X^{(s)} = x)$, namely, by the probability that the process is in state y at time $t+s$ given that is was in state x at time s, for all $0 \leq s < t$ and $x, y \in S$. A CTMP is called *time homogeneous* if these conditional probabilities do not depend on s but only on the length of the time interval t, thus, the distribution of the process is determined by the *Markov transition functions*,

$$p_{x,y}(t) \equiv \Pr(X^{(t+s)} = y | X^{(s)} = x), \qquad \text{for all } x, y \in S \text{ and } t \geq 0,$$

which for every fixed t can be viewed as the entries of a stochastic matrix indexed by states x and y.

Under mild assumptions on the Markov transition functions $p_{x,y}(t)$, these functions are differentiable. Their derivatives at $t = 0$,

$$q_{x,y} = \lim_{t \to 0^+} \frac{p_{x,y}(t) - \mathbf{1}_{x=y}}{t},$$

are the entries of the *rate matrix* $\mathbb{Q}$, where $\mathbf{1}$ is the indicator function. This rate matrix describes the infinitesimal transition probabilities,

$$p_{x,y}(h) = \mathbf{1}_{x=y} + q_{x,y} h + o(h), \tag{1}$$

where $o(\cdot)$ means decay to zero faster than its argument, that is $\lim_{h \downarrow 0} \frac{o(h)}{h} = 0$. Note that the off-diagonal entries of $\mathbb{Q}$ are non-negative, whereas each of its rows sums up to zero, namely,

$$q_{x,x} = -\sum_{y \neq x} q_{x,y}.$$

The derivative of the Markov transition function for t other than 0 satisfies the so-called *forward*, or *master equation*,

$$\frac{d}{dt} p_{x,y}(t) = \sum_z q_{z,y} p_{x,z}(t). \tag{2}$$

A similar characterization for the time-dependent probability distribution, $p(t)$, whose entries are defined by

$$p_x(t) = \Pr(X^{(t)} = x), \qquad x \in S,$$

is obtained by multiplying the Markov transition function by entries of the initial distribution $p(0)$ and marginalizing, resulting in

$$\frac{d}{dt} p = p\mathbb{Q}. \tag{3}$$

The solution of this ODE is

$$p(t) = p(0) \exp(t\mathbb{Q}),$$

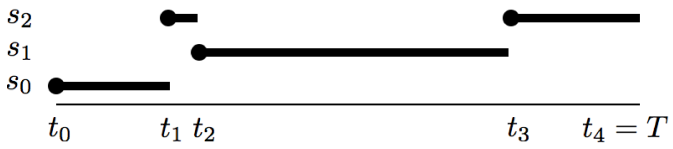


Figure 1: An example of a CTMP trajectory: The process starts at state $x_1 = s_0$, transitions to $x_2 = s_2$ at t_1, to $x_3 = s_1$ at t_2, and finally to $x_4 = s_2$ at t_3.

where $\exp(t\mathbb{Q})$ is a matrix exponential, defined for any square matrix $\mathbb{A}$ by the Taylor series,

$$\exp(\mathbb{A}) = \mathbf{I} + \sum_{k=1}^{\infty} \frac{\mathbb{A}^k}{k!} \ .$$

Applying this solution to the initial condition $p_{x'}(0) = \mathbf{1}_{x=x'}$, we can express the Markov transition function $p_{x,y}(t)$ using the rate matrix $\mathbb{Q}$ as

$$p_{x,y}(t) = [\exp(t\mathbb{Q})]_{x,y}. \tag{4}$$

Although a CTMP is an uncountable collection of random variables (the state of the system at every time t), a *trajectory* σ of $\{X^{(t)}\}_{t\geq 0}$ over a time interval $[0,T]$ can be characterized by a finite number of transitions K, a sequence of states $(x_0, x_1, \ldots, x_K)$ and a sequence of transition times $(t_0 = 0, t_1, \ldots, t_K, t_{K+1} = T)$. We denote by $\sigma(t)$ the state at time t, that is, $\sigma(t) = x_k$ for $t_k \leq t < t_{k+1}$. Figure 1 illustrates such a trajectory.

2.2 Multi-component Representation - Continuous-Time Bayesian Networks

Equation (4) indicates that the distribution of a homogeneous Markov process is fully determined by an initial distribution and a single rate matrix $\mathbb{Q}$. However, since the number of states in a D-component Markov Process is exponential in D, an explicit representation of this transition matrix is often infeasible. *Continuous-time Bayesian networks* are a compact representation of Markov processes that satisfy two assumptions. First it is assumed that only one component can change at a time, thus transition rates involving simultaneous changes of two or more components are zero. Second, the transition rate of each component i depends only on the state of some subset of components denoted $\mathbf{Pa}_i \subseteq \{1,\ldots,D\} \setminus \{i\}$ and on its own state. This dependency is represented using a directed graph, where the nodes are indexed by $\{1,\ldots,D\}$ and the parent nodes of i are $\mathbf{Pa}_i$ (Nodelman et al., 2002). With each component i we then associate a conditional rate matrix $\mathbb{Q}^{i|\mathbf{Pa}_i}_{\cdot|u_i}$ for each state u_i of $\mathbf{Pa}_i$. The off-diagonal entries $q^{i|\mathbf{Pa}_i}_{x_i,y_i|u_i}$ represent the rate at which X_i transitions from state x_i to state y_i given that its parents are in state u_i. The diagonal entries are $q^{i|\mathbf{Pa}_i}_{x_i,x_i|u_i} = -\sum_{y_i \neq x_i} q^{i|\mathbf{Pa}_i}_{x_i,y_i|u_i}$, ensuring that each row in each conditional rate matrix sums up to zero. The dynamics of $X^{(t)}$ are defined by a rate matrix $\mathbb{Q}$ with entries $q_{x,y}$, which combines the conditional rate matrices as follows:

$$q_{x,y} = \begin{cases} q^{i|\mathbf{Pa}_i}_{x_i,y_i|u_i} & \delta(x,y) = \{i\} \\ \sum_i q^{i|\mathbf{Pa}_i}_{x_i,x_i|u_i} & x = y \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $\delta(x,y) = \{j | x_j \neq y_j\}$ denotes the set of components in which x differs from y.

To have another perspective on CTBN's, we may consider a discrete-time approximation of the process. Let h be a sampling interval. The subset of random variables $\{X_{t_k} : k \geq 0\}$, where $t_k = kh$, is a discrete-time Markov process over a D-dimensional state-space. *Dynamic Bayesian networks (DBNs)* provide a compact modeling language for such processes, namely the conditional distribution of a DBN $\mathrm{P}_h(X^{(t_k+1)}|X^{(t_k)})$ is factorized into a product of conditional distributions of $X_i^{(t_{k+1})}$ given the state of a subset of $X^{(t_k)} \cup X^{(t_{k+1})}$. When h is sufficiently small, the CTBN can be approximated by a DBN whose parameters depend on the rate matrix $\mathbb{Q}$ of the CTBN ,

$$\mathrm{P}_h(X^{(t_{k+1})} = y | X^{(t_k)} = x) = \prod_{i=1}^{D} \mathrm{P}_h(X_i^{(t_{k+1})} = y_i | X_i^{(t_k)} = x_i, U^{(t_k)} = u_i), \tag{6}$$

where

$$\mathrm{P}_h(X_i^{(t_{k+1})} = y_i | X_i^{(t_k)} = x_i, U^{(t_k)} = u_i) = \mathbf{1}_{x_i = y_i} + q^{i|\mathbf{Pa}_i}_{x_i,y_i|u_i} h. \tag{7}$$

Each such term is the local conditional probability that $X_i^{(t_{k+1})} = y_i$ given the state of X_i and U_i at time t_k. These are valid conditional distributions, because they are non-negative and are normalized, that is

$$\sum_{y_i \in S_i} \left(\mathbf{1}_{x_i = y_i} + q^{i|\mathbf{Pa}_i}_{x_i,y_i|u_i} h \right) = 1$$

for every x_i and u_i. Note that in this discretized process, transition probabilities involving changes in more than one component are $o(h)$, as in the CTBN. Moreover, using Equations (1) and (5) we observe that

$$\Pr(X^{(t_k+1)} = y | X^{(t_k)} = x) = \mathrm{P}_h(X^{(t_k+1)} = y | X^{(t_k)} = x) + o(h).$$

(See Appendix A for detailed derivations). Therefore, the CTBN and the approximating DBN are asymptotically equivalent as $h \to 0$.

Example 1 An example of a multi-component process is the *dynamic Ising model*, which corresponds to a CTBN in which every component can be in one of two states, -1 or $+1$, and each component prefers to be in the same state as its neighbor. These models are governed by two parameters: a *coupling parameter* β (it is the inverse temperature in physical models, which determines the strength of the coupling between two neighboring components), and a *rate parameter* τ, which determines the propensity of each component to change its state. Low values of β correspond to weak coupling (high temperature). More formally, we define the conditional rate matrices as

$$q^{i|\mathbf{Pa}_i}_{x_i,y_i|u_i} = \tau \left(1 + e^{-2y_i \beta \sum_{j \in \mathbf{Pa}_i} x_j} \right)^{-1}$$

where $x_j \in \{-1, 1\}$. This model is derived by plugging the Ising grid to *Continuous-Time Markov Networks*, which are the undirected counterparts of CTBNs (El-Hay et al., 2006).

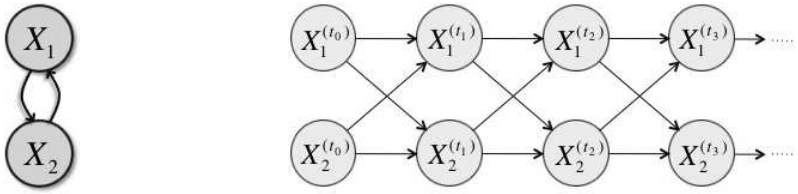
Consider a two component Ising model whose structure and corresponding DBN are shown in Figure 2. This system is symmetric, that is, the conditional rate matrices are identical for $i \in \{1,2\}$. As an example, for a specific choice of β and τ we have:

$\mathbb{Q}^{i|\mathbf{Pa}_i}_{\cdot|-1}$ =

	-	+
-	-1	1
+	10	-10

$\mathbb{Q}^{i|\mathbf{Pa}_i}_{\cdot|+1}$ =

	-	+
-	-10	10
+	1	-1



(a) (b)

Figure 2: Two representations of a two binary component dynamic process. *(a)* The associated CTBN. *(b)* The DBN corresponding to the CTBN in (a). The models are equivalent when $h \to 0$.

The local conditional distributions of the DBN can be directly inferred from Equation (7). For example

$$\mathrm{P}_h(X_1^{(t_{k+1})} = 1 | X_1^{(t_k)} = -1, X_2^{(t_k)} = 1) = 10h.$$

Here, in both components the conditional rates are higher for transitions into states that are identical to the state of their parent. Therefore, the two components have a disposition of being in the same state. To support this intuition, we examine the amalgamated rate matrix:

$$\mathbb{Q} = \begin{array}{c|cccc} & \text{--} & \text{-+} & \text{+-} & \text{++} \\ \hline \text{--} & -2 & 1 & 1 & 0 \\ \text{-+} & 10 & -20 & 0 & 10 \\ \text{+-} & 10 & 0 & -20 & 10 \\ \text{++} & 0 & 1 & 1 & -2. \end{array}$$

Clearly, transition rates into states in which both components have the same value is higher. Higher transitions rate imply higher transition probabilities, for example:

$$\begin{aligned} p_{\text{-+},\text{--}}(h) &= 10h + o(h), \\ p_{\text{--},\text{-+}}(h) &= h + o(h). \end{aligned}$$

Thus the probability of transitions into a coherent state is much higher than into an incoherent state. ∎

2.3 Inference in Continuous-time Markov Processes

Our setting is as follows: we receive evidence of the states of several or all components within a time interval $[0,T]$. The two possible types of evidence that may be given are continuous evidence, where we know the state of a subset $U \subseteq X$ continuously over some sub-interval $[t_1,t_2] \subseteq [0,T]$, and point evidence of the state of U at some point $t \in [0,T]$. For convenience we restrict our treatment to a time interval $[0,T]$ with full end-point evidence $X^{(0)} = e_0$ and $X^{(T)} = e_T$. We shall discuss the more general case in Section 5.

Given a continuous-time Bayesian network and evidence of the above type we would like to evaluate the likelihood of the evidence, $\Pr(e_0, e_T; \mathbb{Q})$ and to compute pointwise posterior probabilities of various events (e.g., $\Pr(U^{(t)} = u | e_0, e_T)$ for some $U \subseteq X$). Another class of queries are

conditional expectations of statistics that involve entire trajectories of the process. Two important examples for queries are the *sufficient statistics* required for learning. These statistics are the amount of time in which X_i is in state x_i and $\mathbf{Pa}_i$ are in state u_i, and the number of transitions that X_i underwent from x_i to y_i while its parents were in state u_i (Nodelman et al., 2003). We denote these statistics by $T^i_{x_i|u_i}$ and $M^i_{x_i,y_i|u_i}$ respectively. For example, in the trajectory of the univariate process in Figure 1, we have $T_{s_2} = t_2 - t_1 + t_4 - t_3$ and $M_{s_0,s_2} = 1$.

Exact calculation of these values is usually a computationally intractable task. For instance, calculation of marginals requires first calculating the pointwise distribution over X using a forward-backward like calculation:

$$\Pr(X^{(t)} = x | e_0, e_T) = \frac{p_{e_0,x}(t)\, p_{x,e_T}(T-t)}{p_{e_0,e_T}(T)}, \tag{8}$$

and then marginalizing

$$\Pr(U^{(t)} = u | e_0, e_T) = \sum_{x \setminus u} \Pr(X^{(t)} = x | e_0, e_T),$$

where $p_{x,y}(t) = [\exp(t\mathbb{Q})]_{x,y}$, and the size of $\mathbb{Q}$ is exponential in the number of components. Moreover, calculating expected residence times and expected number of transitions involves integration over the time interval of these quantities (Nodelman et al., 2005a):

$$\mathbf{E}[T_x] = \frac{1}{p_{e_0,e_T}(T)} \int_0^T p_{e_0,x}(t)\, p_{x,e_T}(T-t)dt,$$
$$\mathbf{E}[M_{x,y}] = \frac{1}{p_{e_0,e_T}(T)} \int_0^T p_{e_0,x}(t)\, q_{x,y}\, p_{y,e_T}(T-t)dt \quad .$$

These make this approach infeasible beyond a modest number of components, hence we have to resort to approximations.

3. Variational Principle for Continuous-Time Markov Processes

Variational approximations to structured models aim to approximate a complex distribution by a simpler one, which allows efficient inference. This problem can be viewed as an optimization problem: given a specific model and evidence, find the "best" approximation within a given class of simpler distributions. In this setting the inference is posed as a constrained optimization problem, where the constraints ensure that the parameters correspond to valid distributions consistent with the evidence. Specifically, the optimization problem is constructed by defining a lower bound to the log-likelihood of the evidence, where the gap between the bound and the true likelihood is the divergence of between the approximation and the true posterior. While the resulting problem is generally intractable, it enables us to derive approximate algorithms by approximating either the functional or the constrains that define the set of valid distributions. In this section we define the lower-bound functional in terms of a general continuous-time Markov process (that is, without assuming any network structure). Here we aim at defining a lower bound on $\ln P_{\mathbb{Q}}(e_T|e_0)$ as well as to approximating the posterior probability $P_{\mathbb{Q}}(\cdot \mid e_0, e_T)$, where $P_{\mathbb{Q}}$ is the distribution of the Markov process whose instantaneous rate-matrix is $\mathbb{Q}$. We start by examining the structure of the posterior and introducing an appropriate parameterization.

Recall that the distribution of a time-homogeneous Markov process is characterized by the conditional transition probabilities $p_{x,y}(t)$, which in turn is fully redetermined by the constant rate matrix $\mathbb{Q}$. It is not hard to see that whenever the prior distribution of a stochastic process is that of a homogeneous Markov process with rate matrix $\mathbb{Q}$, then the posterior $P_{\mathbb{Q}}(\cdot|e_0, e_T)$ is also a Markov process, albeit generally not a homogeneous one. The distribution of a continuous-time Markov processes that is not homogeneous in time is determined by conditional transition probabilities, $p_{x,y}(s, s+t)$, which depend explicitly on both initial and final times. These transition probabilities can be expressed by means of a time-dependent matrix-valued function, $\mathbb{R}(t)$, which describes instantaneous transition rates. The connection between the time-dependent rate matrix $\mathbb{R}(t)$ and the transition probabilities, $p_{x,y}(s, s+t)$ is established by the master equation,

$$\frac{d}{dt} p_{x,y}(s, s+t) = \sum_z r_{z,y}(s+t) p_{x,z}(s, s+t),$$

where $r_{z,y}(t)$ are the entries of $\mathbb{R}(t)$. This equation is a generalization of Equation (2) for inhomogeneous processes. As in the homogeneous case, it leads to a master equation for the time-dependent probability distribution,

$$\frac{d}{dt} p_x(t) = \sum_y r_{y,x}(t) p_y(t),$$

thereby generalizing Equation (3).

By the above discussion, it follows that the posterior process can be represented by a time-dependent rate matrix $\mathbb{R}(t)$. More precisely, writing the posterior transition probability using basic properties of conditional probabilities and the definition of the Markov transition function gives

$$P_{\mathbb{Q}}(X^{(t+h)} = y | X^{(t)} = x, X^{(T)} = e_T) = \frac{p_{x,y}(h) p_{y,e_T}(T - t + h)}{p_{x,e_T}(T - t)}.$$

Taking the limit $h \to 0$ we obtain the instantaneous transition rate of the posterior process

$$r_{x,y}(t) = \lim_{h \to 0} \frac{P_{\mathbb{Q}}(X^{(t+h)} = y | X^{(t)} = x, X^{(T)} = e_T)}{h} = q_{x,y} \cdot \frac{p_{y,e_T}(T - t)}{p_{x,e_T}(T - t)}. \tag{9}$$

This representation, although natural, proves problematic in the framework of deterministic evidence because as t approaches T the transition rate into the observed state tends to infinity. In particular, when $x \neq e_T$ and $y = e_T$, the posterior transition rate is $q_{x,e_T} \cdot \frac{p_{e_T,e_T}(T-t)}{p_{x,e_T}(T-t)}$. This term diverges as $t \to T$, because the numerator approaches 1 while the denominator approaches 0. We therefore consider an alternative parameterization for this inhomogeneous process that is more suitable for variational approximations.

3.1 Marginal Density Representation

Let Pr be the distribution of a Markov process, generally not time homogeneous. We define a family of functions:

$$\begin{aligned} \mu_x(t) &= \Pr(X^{(t)} = x), \\ \gamma_{x,y}(t) &= \lim_{h \downarrow 0} \frac{\Pr(X^{(t)} = x, X^{(t+h)} = y)}{h}, \ x \neq y. \end{aligned} \tag{10}$$

The function $\mu_x(t)$ is the marginal probability that $X^{(t)} = x$. The function $\gamma_{x,y}(t)$ is the probability density that X transitions from state x to y at time t. Note that this parameter is not a transition rate, but rather a product of a point-wise probability with the point-wise transition rate of the distribution, that is, the entries of the time-dependent rate matrix of an equivalent process can be defined by

$$r_{x,y}(t) = \begin{cases} \frac{\gamma_{x,y}(t)}{\mu_x(t)} & \mu_x(t) > 0, \\ 0 & \mu_x(t) = 0. \end{cases} \tag{11}$$

Hence, unlike the (inhomogeneous) rate matrix at time t, $\gamma_{x,y}(t)$ takes into account the probability of being in state x and not only the rate of transitions.

We aim to use the family of functions μ and γ as a representation of the posterior process. To do so, we need to characterize the set of constraints that these functions satisfy. We begin by constraining the marginals $\mu_x(t)$ to be valid distributions that is, $0 \leq \mu_x(t) \leq 1$ and $\sum_x \mu_x(t) = 1$. A similar constraint on the pairwise distributions implies that $\gamma_{x,y}(t) \geq 0$ for $x \neq y$. Next, we infer additional constraints from consistency properties between distributions over pairs of variables and their uni-variate marginals. Specifically, Equation (10) implies that for $x \neq y$

$$\Pr(X^{(t)} = x, X^{(t+h)} = y) = \gamma_{x,y}(t)\, h + o(h). \tag{12}$$

Plugging this identity into the consistency constraint

$$\mu_x(t) = \Pr(X^{(t)} = x) = \sum_y \Pr(X^{(t)} = x, X^{(t+h)} = y),$$

defining

$$\gamma_{x,x}(t) = -\sum_{y \neq x} \gamma_{x,y}(t)$$

and rearranging, we obtain

$$\Pr(X^{(t)} = x, X^{(t+h)} = y) = \mathbf{1}_{x=y}\mu_x(t) + \gamma_{x,y}(t)\, h + o(h), \tag{13}$$

which unlike (12) is valid for all x, y. Marginalizing (13) with respect to the second variable,

$$\Pr(X^{(t+h)} = x) = \sum_y \Pr(X^{(t)} = y, X^{(t+h)} = x),$$

we obtain a forward update rule for the uni-variate marginals

$$\mu_x(t+h) = \mu_x(t) + h \sum_y \gamma_{y,x}(t) + o(h).$$

Rearranging terms and taking the limit $h \to 0$ gives a differential equation for $\mu_x(t)$,

$$\frac{d}{dt}\mu_x(t) = \sum_y \gamma_{y,x}(t).$$

Finally, whenever $\mu_x(t) = 0$ we have $\Pr(X^{(t)} = x, X^{(t+h)} = y) = 0$, implying in that case that $\gamma_{x,y}(t) = 0$. Based on these observations we define:

Definition 1 A family $\eta = \{\mu_x(t), \gamma_{x,y}(t) : 0 \le t \le T\}$ of functions is a *Markov-consistent density set* if the following constraints are fulfilled:

$$\begin{aligned} \mu_x(t) &\geq 0, \quad \sum_x \mu_x(0) = 1, \\ \gamma_{x,y}(t) &\geq 0 \qquad \forall y \neq x, \\ \gamma_{x,x}(t) &= -\sum_{y \neq x} \gamma_{x,y}(t), \\ \frac{d}{dt}\mu_x(t) &= \sum_y \gamma_{y,x}(t), \end{aligned}$$

and $\gamma_{x,y}(t) = 0$ whenever $\mu_x(t) = 0$. We denote by $\mathcal{M}$ the set of all Markov-consistent densities. ∎

Using standard arguments we can show that there exists a correspondence between (generally inhomogeneous) Markov processes and density sets η. Specifically, given η, we construct a process by defining an inhomogeneous rate matrix $\mathbb{R}(t)$ whose entries are defined in Equation (11) and prove the following:

Lemma 2 *Let $\eta = \{\mu_x(t), \gamma_{x,y}(t) : 0 \le t \le T\}$. If $\eta \in \mathcal{M}$, then there exists a continuous-time Markov process* Pr *for which μ_x and $\gamma_{x,y}$ satisfy (10) for every t in the right-open interval [0,T).*

Proof See appendix B ■

The converse is also true: for every integrable inhomogeneous rate matrix $\mathbb{R}(t)$ the corresponding marginal density set is defined by $\frac{d}{dt}\mu_x(t) = \sum_y r_{y,x}(t)\mu_y(t)$ and $\gamma_{x,y}(t) = \mu_x(t)r_{x,y}(t)$. The processes we are interested in, however, have additional structure, as they correspond to the posterior distribution of a time-homogeneous process with end-point evidence. In that case, multiplying Equation (9) by $\mu_x(t)$ gives

$$\gamma_{x,y}(t) = \mu_x(t) \cdot q_{x,y} \cdot \frac{p_{y,e_T}(T-t)}{p_{x,e_T}(T-t)}. \tag{14}$$

Plugging in Equation (8) we obtain

$$\gamma_{x,y}(t) = \frac{p_{e_0,x}(t) \cdot q_{x,y} \cdot p_{y,e_T}(T-t)}{p_{e_0,e_T(T)}},$$

which is zero when $y \neq e_T$ and $t = T$. This additional structure implies that we should only consider a subset of $\mathcal{M}$. Specifically the representation η corresponding to the posterior distribution $P_{\mathbb{Q}}(\cdot|e_0, e_T)$ satisfies $\mu_x(0) = \mathbf{1}_{x=e_0}, \mu_x(T) = \mathbf{1}_{x=e_T}, \gamma_{x,y}(0) = 0$ for all $x \neq e_0$ and $\gamma_{x,y}(T) = 0$ for all $y \neq e_T$. We denote by $\mathcal{M}_e \subset \mathcal{M}$ the subset that contains Markov-consistent density sets satisfying these constraints. This analysis suggests that for every homogeneous rate matrix and point evidence e there is a member in $\mathcal{M}_e$ that corresponds to the posterior process. Thus, from now on we restrict our attention to density sets from $\mathcal{M}_e$.

3.2 Variational Principle

The marginal density representation allows us to state the variational principle for continuous processes, which closely tracks similar principles for discrete processes. Specifically, we define a functional of functions that are constrained to be density sets from $\mathcal{M}_e$. The maximum over this

set is the log-likelihood of the evidence and is attained for a density set that represents the posterior distribution. This formulation will serve as a basis for the mean-field approximation, which is introduced in the next section.

Define a *free energy functional*,

$$\mathcal{F}(\eta;\mathbb{Q}) = \mathcal{E}(\eta;\mathbb{Q}) + \mathcal{H}(\eta),$$

which, as we will see, measures the quality of η as an approximation of $P_{\mathbb{Q}}(\cdot|e)$. (For succinctness, we will assume that the evidence e is clear from the context.) The two terms in the functional are the *average energy*,

$$\mathcal{E}(\eta;\mathbb{Q}) = \int_0^T \sum_x \left[\mu_x(t) q_{x,x} + \sum_{y \neq x} \gamma_{x,y}(t) \ln q_{x,y} \right] dt,$$

and the *entropy*,

$$\mathcal{H}(\eta) = \int_0^T \sum_x \sum_{y \neq x} \gamma_{x,y}(t)[1 + \ln \mu_x(t) - \ln \gamma_{x,y}(t)]dt.$$

The following theorem establishes the relation of this functional to the Kullback-Leibler (KL) divergence and the likelihood of the evidence, and thus allows us to cast the variational inference into an optimization problem.

Theorem 3 *Let $\mathbb{Q}$ be a rate matrix, $e = (e_0, e_T)$ be states of X, and $\eta \in \mathcal{M}_e$. Then*

$$\mathcal{F}(\eta;\mathbb{Q}) = \ln P_{\mathbb{Q}}(e_T|e_0) - \boldsymbol{D}(P_\eta \| P_{\mathbb{Q}}(\cdot|e))$$

where P_η is the distribution corresponding to η and $\boldsymbol{D}(P_\eta \| P_{\mathbb{Q}}(\cdot|e))$ is the KL divergence between the two processes.

We conclude from the non-negativity of the KL divergence that the energy functional $\mathcal{F}(\eta;\mathbb{Q})$ is a lower bound of the log-likelihood of the evidence. The closer the approximation to the target posterior, the tighter the bound. Moreover, since the KL divergence is zero if and only if the two distributions are equal almost everywhere, finding the maximizer of this free energy is equivalent to finding the posterior distribution from which answers to different queries can be efficiently computed.

3.3 Proof of Theorem 3

We begin by examining properties of distributions of inhomogeneous Markov processes. Let $X^{(t)}$ be an inhomogeneous Markov process with rate matrix $\mathbb{R}(t)$. As in the homogeneous case, a trajectory σ of $\{X^{(t)}\}_{t \geq 0}$ over a time interval $[0,T]$ can be characterized by a finite number of transitions K, a sequence of states $(x_0, x_1, \ldots, x_K)$ and a sequence of transition times $(t_0 = 0, t_1, \ldots, t_K, t_{K+1} = T)$. We denote by Σ the set of all trajectories of $X^{[0,T]}$. The distribution over Σ can be characterized by a collection of random variables that consists of the number of transitions κ, a sequence of states $(\chi_0, \ldots, \chi_\kappa)$ and transition times $(\tau_1, \ldots, \tau_\kappa)$. Note that the number of random variables that characterize the trajectory is by itself a random variable. The density $f_{\mathbb{R}}$ of a trajectory $\sigma = \{K, x_0, \ldots, x_K, t_1, \ldots, t_K\}$ is the derivative of the joint distribution with respect to transition times, that is,

$$f_{\mathbb{R}}(\sigma) = \frac{\partial^K}{\partial t_1 \cdots \partial t_K} P_{\mathbb{R}}(\kappa = K, \chi_0 = x_0, \ldots, \chi_K = x_K, \tau_1 \leq t_1, \ldots, \tau_K \leq t_K),$$

which is given by

$$f_{\mathbb{R}}(\sigma) = p_{x_0}(0) \cdot \prod_{k=0}^{K-1} \left[e^{\int_{t_k}^{t_{k+1}} r_{x_k,x_k}(t)dt} r_{x_k,x_{k+1}}(t_{k+1}) \right] \cdot e^{\int_{t_K}^{t_{K+1}} r_{x_K,x_K}(t)dt}.$$

For example, in case $\mathbb{R}(t) = \mathbb{Q}$ is a homogeneous rate matrix this equation reduces to

$$f_{\mathbb{Q}}(\sigma) = p_{x_0}(0) \cdot \prod_{k=0}^{K-1} \left[e^{q_{x_k,x_k}(t_{k+1}-t_k)} q_{x_k,x_{k+1}} \right] \cdot e^{q_{x_K,x_K}(t_{K+1}-t_K)}.$$

The expectation of a random variable $\psi(\sigma)$ is an infinite sum (because one has to account for all possible numbers of transitions) of finite dimensional integrals,

$$\mathbf{E}_{f_{\mathbb{Q}}}[\psi] \equiv \int_{\Sigma} f_{\mathbb{R}}(\sigma)\psi(\sigma)d\sigma \equiv \sum_{K=0}^{\infty} \sum_{x_0} \cdots \sum_{x_K} \int_0^T \int_0^{t_K} \cdots \int_0^{t_2} f_{\mathbb{R}}(\sigma)\psi(\sigma)dt_1 \cdots dt_K.$$

The *KL-divergence* between two densities that correspond to two inhomogeneous Markov processes with rate matrices $\mathbb{R}(t)$ and $\mathbb{S}(t)$ is

$$\boldsymbol{D}(f_{\mathbb{R}} \| f_{\mathbb{S}}) = \int_{\Sigma} f_{\mathbb{R}}(\sigma) \ln \frac{f_{\mathbb{R}}(\sigma)}{f_{\mathbb{S}}(\sigma)} d\sigma \ . \tag{15}$$

We will use the convention $0 \ln 0 = 0$ and assume the support of $f_{\mathbb{S}}$ is contained in the support of $f_{\mathbb{R}}$. That is $f_{\mathbb{R}}(\sigma) = 0$ whenever $f_{\mathbb{S}}(\sigma) = 0$. The KL-divergence satisfies $\boldsymbol{D}(f_{\mathbb{R}} \| f_{\mathbb{S}}) \geq 0$ and is exactly zero if and only if $f_{\mathbb{R}} = f_{\mathbb{S}}$ almost everywhere (Kullback and Leibler, 1951).

Let $\eta \in \mathcal{M}_e$ be a marginal density set consistent with e. As we have seen, this density set corresponds to a Markov process with rate matrix $\mathbb{R}(t)$ whose entries are defined by Equation (11), hence we identify $f_\eta \equiv f_{\mathbb{R}}$.

Given evidence e on some event we denote $f_{\mathbb{Q}}(\sigma, e) \equiv f_{\mathbb{Q}}(\sigma) \cdot \mathbf{1}_{\sigma \models e}$, and note that

$$P_{\mathbb{Q}}(e) = \int_{\{\sigma : \sigma \models e\}} f_{\mathbb{Q}}(\sigma)d\sigma = \int_{\Sigma} f_{\mathbb{Q}}(\sigma, e)d\sigma \ ,$$

where $\sigma \models e$ is a predicate which is true if σ is consistent with the evidence. The density function of the posterior distribution $P_{\mathbb{Q}}(\cdot|e)$ satisfies $f_{\mathbb{S}}(\sigma) = \frac{f_{\mathbb{Q}}(\sigma,e)}{P_{\mathbb{Q}}(e)}$ where $\mathbb{S}(t)$ is the time-dependent rate matrix that corresponds to the posterior process.

Manipulating (15), we get

$$\boldsymbol{D}(f_\eta \| f_{\mathbb{S}}) = \int_{\Sigma} f_\eta(\sigma) \ln f_\eta(\sigma) d\sigma - \int_{\Sigma} f_\eta(\sigma) \ln f_{\mathbb{S}}(\sigma) d\sigma \equiv \mathbf{E}_{f_\eta}[\ln f_\eta(\sigma)] - \mathbf{E}_{f_\eta}[\ln f_{\mathbb{S}}(\sigma)].$$

Replacing $\ln f_{\mathbb{S}}(\sigma)$ by $\ln f_{\mathbb{Q}}(\sigma, e) - \ln P_{\mathbb{Q}}(e)$ and applying simple arithmetic operations gives

$$\ln P_{\mathbb{Q}}(e) = \mathbf{E}_{f_\eta}[\ln f_{\mathbb{Q}}(\sigma, e)] - \mathbf{E}_{f_\eta}[\ln f_\eta(\sigma)] + \boldsymbol{D}(f_\eta \| f_{\mathbb{S}}).$$

The crux of the proof is in showing that the expectations in the right-hand side satisfy

$$\mathbf{E}_{f_\eta}[\ln f_{\mathbb{Q}}(\sigma, e)] = \mathcal{E}(\eta; \mathbb{Q}),$$

and

$$-\mathbf{E}_{f_\eta}\left[\ln f_\eta(\sigma)\right] = \mathcal{H}(\eta),$$

implying that $\mathcal{F}(\eta;\mathbb{Q})$ is a lower bound on the log-probability of evidence with equality if and only if $f_\eta = f_\mathbb{Q}$ almost everywhere.

To prove these identities for the energy and entropy, we treat trajectories as functions $\sigma : \mathcal{R} \to \mathcal{R}$ where $\mathcal{R}$ is the set of real numbers by denoting $\sigma(t) \equiv X^{(t)}(\sigma)$—the state of the system at time t. Using this notation we introduce two lemmas that allow us to replace integration over a set of trajectories by a one dimensional integral, which is defined over a time variable. The first result handles expectations of functions that depend on specific states:

Lemma 4 *Let $\psi : S \times \mathcal{R} \to \mathcal{R}$ be a function, then*

$$\mathbf{E}_{f_\eta}\left[\int_0^T \psi(\sigma(t),t)dt\right] = \int_0^T \sum_x \mu_x(t)\psi(x,t)dt.$$

Proof See Appendix C.1 ■

As an example, by setting $\psi(x',t) = \mathbf{1}_{x'=x}$ we obtain that the expected residence time in state x is $\mathbf{E}_{f_\eta}[T_x] = \int_0^T \mu_x(t)dt$. The second result handles expectations of functions that depend on transitions between states:

Lemma 5 *Let $\psi(x,y,t)$ be a function from $S \times S \times \mathcal{R}$ to $\mathcal{R}$ that is continuous with respect to t and satisfies $\psi(x,x,t) = 0$, $\forall x, \forall t$ then*

$$\mathbf{E}_{f_\eta}\left[\sum_{k=1}^{K^\sigma} \psi(x_{k-1}^\sigma, x_k^\sigma, t_k^\sigma)\right] = \int_0^T \sum_x \sum_{y\neq x} \gamma_{x,y}(t)\psi(x,y,t)dt,$$

where the superscript σ stresses that K^σ, x_k^σ and t_k^σ are associated with a specific trajectory σ.

Proof See Appendix C.2 ■

Continuing the example of the previous lemma, here by setting $\psi(x',y',t) = \mathbf{1}_{x'=x}\mathbf{1}_{y'=y}\mathbf{1}_{x\neq y}$ the sums within the left hand expectation become the number of transitions in a trajectory σ. Thus, we obtain that the expected number of transitions from x to y is $\mathbf{E}_f[M_{x,y}] = \int_0^T \gamma_{x,y}(t)dt$.

We now use these lemmas to compute the expectations involved in the energy functional. Suppose $e = \{e_0, e_T\}$ is a pair of point evidence and $\eta \in \mathcal{M}_e$. Applying these lemmas with $\psi(x,t) = q_{x,x}$ and $\psi(x,y,t) = \mathbf{1}_{x\neq y} \cdot \ln q_{x,y}$ gives

$$\mathbf{E}_{f_\eta}\left[\ln f_\mathbb{Q}(\sigma,e)\right] = \int_0^T \sum_x \left[\mu_x(t)q_{x,x}(t) + \sum_{y\neq x} \gamma_{x,y}(t)\ln q_{x,y}(t)\right] dt\ .$$

Similarly, setting $\psi(x,t) = r_{x,x}(t)$ and $\psi(x,y,t) = \mathbf{1}_{x\neq y} \cdot \ln r_{x,y}(t)$, where $\mathbb{R}(t)$ is defined in Equation (11), we obtain

$$-\mathbf{E}_{f_\eta}\left[\ln f_\eta(\sigma,e)\right] = -\int_0^T \sum_x \left[\mu_x(t)\frac{\gamma_{x,x}(t)}{\mu_x(t)} + \sum_{y\neq x} \gamma_{x,y}(t)\ln\frac{\gamma_{x,y}(t)}{\mu_x(t)}\right] dt = \mathcal{H}(\eta)\ .$$

4. Factored Approximation

The variational principle we discussed is based on a representation that is as complex as the original process—the number of functions $\gamma_{x,y}(t)$ we consider is equal to the size of the original rate matrix $\mathbb{Q}$. To get a tractable inference procedure we make additional simplifying assumptions on the approximating distribution.

Given a D-component process we consider approximations that factor into products of independent processes. More precisely, we define $\mathcal{M}_e^i$ to be the continuous Markov-consistent density sets over the component X_i, that are consistent with the evidence on X_i at times 0 and T. Given a collection of density sets $\eta^1,\ldots,\eta^D$ for the different components, the product density set $\eta = \eta^1 \times \cdots \times \eta^D$ is defined as

$$\begin{aligned} \mu_x(t) &= \prod_i \mu^i_{x_i}(t), \\ \gamma_{x,y}(t) &= \begin{cases} \gamma^i_{x_i,y_i}(t)\mu_x^{\setminus i}(t) & \delta(x,y) = \{i\} \\ \sum_i \gamma^i_{x_i,x_i}(t)\mu_x^{\setminus i}(t) & x = y \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where $\mu_x^{\setminus i}(t) = \prod_{j\neq i}\mu^j_{x_j}(t)$ is the joint distribution at time t of all the components other than the i'th (it is not hard to see that if $\eta^i \in \mathcal{M}_e^i$ for all i, then $\eta \in \mathcal{M}_e$). We define the set $\mathcal{M}_e^F$ to contain all factored density sets. From now on we assume that $\eta = \eta^1 \times \cdots \times \eta^D \in \mathcal{M}_e^F$.

Assuming that $\mathbb{Q}$ is defined by a CTBN, and that η is a factored density set, we can rewrite

$$\mathcal{E}(\eta;\mathbb{Q}) = \sum_i \int_0^T \sum_{x_i} \left[\mu^i_{x_i}(t)\mathbf{E}_{\mu^{\setminus i}(t)}\left[q_{x_i,x_i|U_i}\right] + \sum_{x_i,y_i\neq x_i} \gamma^i_{x_i,y_i}(t)\mathbf{E}_{\mu^{\setminus i}(t)}\left[\ln q_{x_i,y_i|U_i}\right] \right] dt$$

(see derivations in Appendix D). Similarly, the entropy term factors as

$$\mathcal{H}(\eta) = \sum_i \mathcal{H}(\eta^i) \ .$$

Note that terms such as $\mathbf{E}_{\mu^{\setminus i}(t)}\left[q_{x_i,x_i|U_i}\right]$ involve only $\mu^j(t)$ for $j \in \mathbf{Pa}_i$, because $\mathbf{E}_{\mu^{\setminus i}(t)}[f(U_i)] = \sum_{u_i}\mu_{u_i}(t)f(u_i)$. Therefore, this decomposition involves only local terms that either include the i'th component, or include the i'th component and its parents in the CTBN defining $\mathbb{Q}$.

To make the factored nature of the approximation explicit in the notation, we write henceforth,

$$\mathcal{F}(\eta;\mathbb{Q}) = \tilde{\mathcal{F}}(\eta^1,\ldots,\eta^D;\mathbb{Q}).$$

4.1 Fixed Point Characterization

The factored form of the functional and the independence between the different η^i allows optimization by *block ascent*, optimizing the functional with respect to each parameter set in turn. To do so, we should solve the following optimization problem:

> Fixing i, and given $\eta^1,\ldots,\eta^{i-1},\eta^{i+1},\ldots,\eta^D$, in $\mathcal{M}_e^1,\ldots\mathcal{M}_e^{i-1},\mathcal{M}_e^{i+1},\ldots,\mathcal{M}_e^D$, respectively, find
>
> $$\arg\max_{\eta^i\in\mathcal{M}_e^i} \tilde{\mathcal{F}}(\eta^1,\ldots,\eta^D;\mathbb{Q}) \ .$$

If for all i, we have a $\mu^i \in \mathcal{M}_e^i$, which is a solution to this optimization problem with respect to each component, then we have a (local) stationary point of the energy functional within $\mathcal{M}_e^F$.

To solve this optimization problem, we define a Lagrangian, which includes the constraints in the form of Definition 1. These constraints are to be enforced in a continuous fashion, and so the Lagrange multipliers $\lambda^i_{x_i}(t)$ are continuous functions of t as well. The Lagrangian is a functional of the functions $\mu^i_{x_i}(t)$, $\gamma^i_{x_i,y_i}(t)$ and $\lambda^i_{x_i}(t)$, and takes the following form

$$\mathcal{L} = \tilde{\mathcal{F}}(\eta;\mathbb{Q}) - \sum_{i=1}^{D} \int_0^T \lambda^i_{x_i}(t) \left(\frac{d}{dt}\mu^i_{x_i}(t) - \sum_{y_i} \gamma^i_{x_i,y_i}(t) \right) dt \ .$$

A necessary condition for the optimality of a density set η is the existence of λ such that (η,λ) is a *stationary point* of the Lagrangian. A stationary point of a functional satisfies the *Euler-Lagrange* equations, namely the *functional derivatives* with respect to μ, γ and λ vanish (see Appendix E for a brief review). The detailed derivation of the resulting equations is in Appendix F. Writing these equations in explicit form, we get a fixed point characterization of the solution in term of the following set of ODEs:

$$\begin{aligned} \frac{d}{dt}\mu^i_{x_i}(t) &= \sum_{y_i \neq x_i} \left(\gamma^i_{y_i,x_i}(t) - \gamma^i_{x_i,y_i}(t) \right), \\ \frac{d}{dt}\rho^i_{x_i}(t) &= -\rho^i_{x_i}(t) \left(\overline{q}^{\,i}_{x_i,x_i}(t) + \psi^i_{x_i}(t) \right) - \sum_{y_i \neq x_i} \rho^i_{y_i}(t) \tilde{q}^i_{x_i,y_i}(t) \end{aligned} \tag{16}$$

along with the following algebraic constraint

$$\rho^i_{x_i}(t)\gamma^i_{x_i,y_i}(t) = \mu^i_{x_i}(t)\tilde{q}^i_{x_i,y_i}(t)\rho^i_{y_i}(t), \ x_i \neq y_i \tag{17}$$

where ρ^i are the exponents of the Lagrange multipliers λ_i. In these equations we use the following shorthand notations for the average rates

$$\begin{aligned} \overline{q}^{\,i}_{x_i,x_i}(t) &= \mathbf{E}_{\mu^{\setminus i}(t)} \left[q^{i|\mathbf{Pa}_i}_{x_i,x_i|U_i} \right], \\ \overline{q}^{\,i}_{x_i,x_i|x_j}(t) &= \mathbf{E}_{\mu^{\setminus i}(t)} \left[q^{i|\mathbf{Pa}_i}_{x_i,x_i|U_i} \mid x_j \right], \end{aligned}$$

where $\mu^{\setminus i}(t)$ is the product distribution of $\mu^1(t),\ldots,\mu^{i-1}(t),\mu^{i+1}(t),\ldots,\mu^D(t)$. Similarly, we have the following shorthand notations for the geometrically-averaged rates,

$$\begin{aligned} \tilde{q}^i_{x_i,y_i}(t) &= \exp\left\{ \mathbf{E}_{\mu^{\setminus i}(t)} \left[\ln q^{i|\mathbf{Pa}_i}_{x_i,y_i|U_i} \right] \right\}, \\ \tilde{q}^i_{x_i,y_i|x_j}(t) &= \exp\left\{ \mathbf{E}_{\mu^{\setminus i}(t)} \left[\ln q^{i|\mathbf{Pa}_i}_{x_i,y_i|U_i} \mid x_j \right] \right\} \ . \end{aligned}$$

The last auxiliary term is

$$\psi^i_{x_i}(t) = \sum_{j \in Children_i} \sum_{x_j} \left[\mu^j_{x_j}(t) \overline{q}^{\,j}_{x_j,x_j|x_i}(t) + \sum_{x_j \neq y_j} \gamma^j_{x_j,y_j}(t) \ln \tilde{q}^j_{x_j,y_j|x_i}(t) \right] \ .$$

To uniquely solve the two differential Equations (16) for $\mu^i_{x_i}(t)$ and $\rho^i_{x_i}(t)$ we need to set boundary conditions. The boundary condition for $\mu^i_{x_i}$ is defined explicitly in $\mathcal{M}_e^F$ as

$$\mu^i_{x_i}(0) = \mathbf{1}_{x_i = e_{i,0}} \ . \tag{18}$$

The boundary condition at T is slightly more involved. The constraints in $\mathcal{M}_e^F$ imply that $\mu^i_{x_i}(T) = \mathbf{1}_{x_i=e_{i,T}}$. As stated in Section 3.1, we have that $\gamma^i_{e_{i,T},x_i}(T) = 0$ when $x_i \neq e_{i,T}$. Plugging these values into (17), and assuming that all elements of $\mathbb{Q}^{i|\mathbf{Pa}_i}$ are non-zero we get that $\rho_{x_i}(T) = 0$ for all $x_i \neq e_{i,T}$ (It might be possible to use a weaker condition that $\mathbb{Q}$ is irreducible). In addition, we notice that $\rho_{e_{i,T}}(T) \neq 0$, for otherwise the whole system of equations for ρ will collapse to 0. Finally, notice that the solution of (16,17) for μ^i and γ^i is insensitive to the multiplication of ρ^i by a constant. Thus, we can arbitrarily set $\rho_{e_{i,T}}(T) = 1$, and get the boundary condition

$$\rho^i_{x_i}(T) = \mathbf{1}_{x_i=e_{i,T}}. \tag{19}$$

Putting it all together we obtain a characterization of stationary points of the functional as stated in the following theorem:

Theorem 6 *$\eta^i \in \mathcal{M}_e^i$ is a stationary point (e.g., local maxima) of $\tilde{\mathcal{F}}(\eta^1,\ldots,\eta^D;\mathbb{Q})$ subject to the constraints of Definition 1 if and only if it satisfies (16–19).*

Proof see Appendix F ■

It is straightforward to extend this result to show that at a maximum with respect to all the component densities, this fixed-point characterization must hold for all components simultaneously.

Example 2 Consider the case of a single component, for which our procedure should be exact, as no simplifying assumptions are made on the density set. In that case, the averaged rates $\overline{q}^i$ and the geometrically-averaged rates $\tilde{q}^i$ both reduce to the unaveraged rates q, and $\psi \equiv 0$. Thus, the system of equations to be solved is

$$\frac{d}{dt}\mu_x(t) = \sum_{y\neq x}\left(\gamma_{y,x}(t) - \gamma_{x,y}(t)\right),$$
$$\frac{d}{dt}\rho_x(t) = -\sum_y q_{x,y}\rho_y(t),$$

along with the algebraic equation

$$\rho_x(t)\gamma_{x,y}(t) = \mu_x(t)q_{x,y}\rho_y(t), \qquad y \neq x.$$

These equations have a simple intuitive interpretation. First, the backward propagation rule for ρ_x implies that

$$\rho_x(t) = \Pr(e_T|X^{(t)} = x).$$

To prove this identity, we recall the notation $p_{x,y}(h) \equiv \Pr(X^{(t+h)} = y|X^{(t)} = x)$ and write the discretized propagation rule

$$\Pr(e_T|X^{(t)} = x) = \sum_y p_{x,y}(h)\cdot\Pr(e_T|X^{(t+h)} = y) \ .$$

Using the definition of q (Equation 1), rearranging, dividing by h and taking the limit $h \to 0$ gives

$$\frac{d}{dt}\Pr(e_T|X^{(t)} = x) = -\sum_y q_{x,y}\cdot\Pr(e_T|X^{(t)} = y),$$

which is identical to the differential equation for ρ. Second, dividing the above algebraic equation by $\rho_x(t)$ whenever it is greater than zero we obtain

$$\gamma_{x,y}(t) = \mu_x(t) q_{x,y} \frac{\rho_y(t)}{\rho_x(t)}. \tag{20}$$

Thus, we reconstructed Equation (14).

This analysis suggest that this system of ODEs is similar to forward-backward propagation, except that unlike classical forward propagation, here the forward propagation already takes into account the backward messages to directly compute the posterior. Given this interpretation, it is clear that integrating $\rho_x(t)$ from T to 0 followed by integrating $\mu_x(t)$ from 0 to T computes the exact posterior of the processes.

This interpretation of $\rho_x(t)$ also allows us to understand the role of $\gamma_{x,y}(t)$. Equation (20) suggests that the instantaneous rate combines the original rate with the relative likelihood of the evidence at T given y and x. If y is much more likely to lead to the final state, then the rates are biased toward y. Conversely, if y is unlikely to lead to the evidence the rate of transitions to it are lower. This observation also explains why the forward propagation of μ_x will reach the observed $\mu_x(T)$ even though we did not impose it explicitly. ∎

Example 3 Let us return to the two-component Ising chain in Example 1 with initial state $X_1^{(0)} = -1$ and $X_2^{(0)} = 1$, and a reversed state at the final time, $X_1^{(T)} = 1$ and $X_2^{(T)} = -1$. For a large value of β, this evidence is unlikely as at both end points the components are in a undesired configurations. The exact posterior is one that assigns higher probabilities to trajectories where one of the components switches relatively fast to match the other, and then toward the end of the interval, they separate to match the evidence. Since the model is symmetric, these trajectories are either ones in which both components are most of the time in state -1, or ones where both are most of the time in state 1 (Figure 3(a)). Due to symmetry, the marginal probability of each component is around 0.5 throughout most of the interval. The variational approximation cannot capture the dependency between the two components, and thus converges to one of two local maxima, corresponding to the two potential subsets of trajectories (Figure 3(b)). Examining the value of ρ^i, we see that close to the end of the interval they bias the instantaneous rates significantly. For example, as t approaches 1, $\rho_1^1(t)/\rho_{-1}^1(t)$ approaches infinity and so does the instantaneous rate $\gamma_{-1,1}^1(t)/\mu_{-1}^1(t)$, thereby forcing X_1 to switch to state 1 (Figure 3(c)).

This example also allows to examine the implications of modeling the posterior by inhomogeneous Markov processes. In principle, we might have used as an approximation Markov processes with homogeneous rates, and conditioned on the evidence. To examine whether our approximation behaves in this manner, we notice that in the single component case we have

$$q_{x,y} = \frac{\rho_x(t)\gamma_{x,y}(t)}{\rho_y(t)\mu_x(t)},$$

which should be constant.

Consider the analogous quantity in the multi-component case: $\tilde{q}^i_{x_i,y_i}(t)$, the geometric average of the rate of X_i, given the probability of parents state. Not surprisingly, this is exactly a mean field approximation, where the influence of interacting components is approximated by their average influence. Since the distribution of the parents (in the two-component system, the other component)

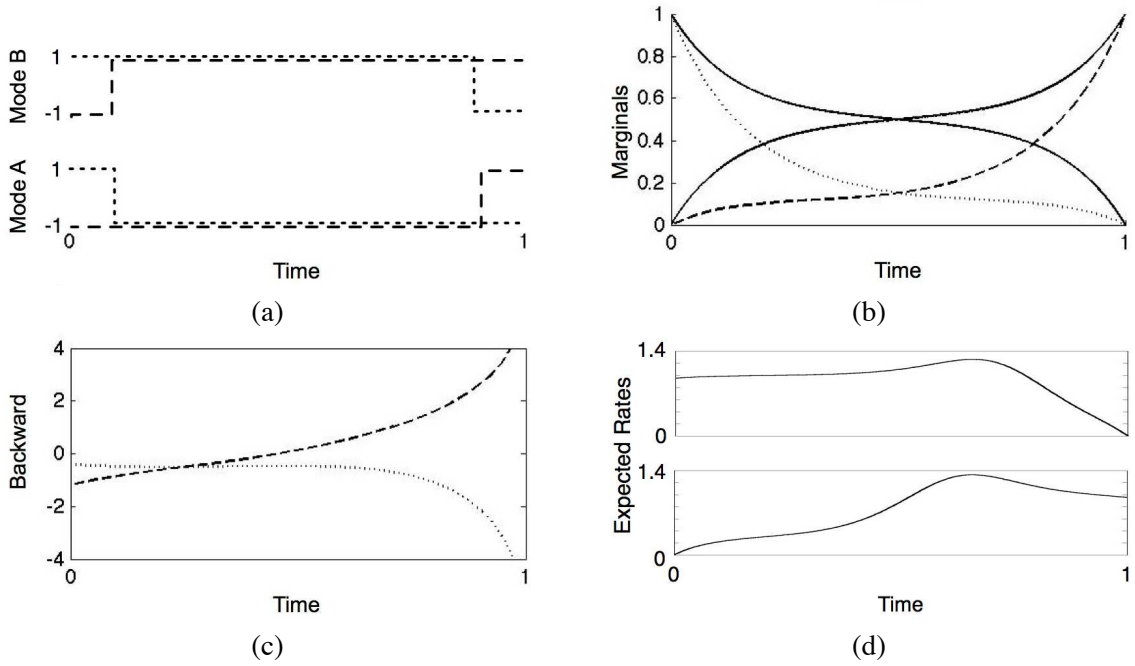


Figure 3: Numerical results for the two-component Ising chain described in Example 3 where the first component starts in state -1 and ends at time $T = 1$ in state 1. The second component has the opposite behavior. *(a)* Two likely trajectories depicting the two modes of the model. *(b)* Exact (solid) and approximate (dashed/dotted) marginals $\mu_1^i(t)$. *(c)* The log ratio $\log \rho_1^i(t)/\rho_{-1}^i(t)$. *(d)* The expected rates $\tilde{q}_{1,-1}^1(t)$ and $\tilde{q}_{-1,1}^1(t)$ of component X_1 of the Ising chain in Example 1. We can notice that the averaged rates are highly non-constant, and so cannot be approximated well with a constant rate matrix.

changes in time, these rates change continuously, especially near the end of the time interval. This suggests that a piecewise homogeneous approximation cannot capture the dynamics without a loss in accuracy. As expected in a dynamic process, we can see in Figure 3(d) that the inhomogeneous transition rates are very erratic. In particular, the rates of X_1 spike at the transition point selected by the mean field approximation. This can be interpreted as putting most of the weight of the distribution on trajectories which transition from state -1 to 1 at that point. ∎

4.2 Optimization Procedure

If $\mathbb{Q}$ is irreducible, then $\rho_{x_i}^i$ and $\mu_{x_i}^i$ are non-zero throughout the open interval $(0,T)$. As a result, we can solve (17) to express γ_{x_i,y_i}^i as a function of μ^i and ρ^i, thus eliminating it from (16) to get evolution equations solely in terms of μ^i and ρ^i. Abstracting the details, we obtain a set of ODEs of the form

$$\frac{d}{dt}\rho^i(t) = \alpha(\rho^i(t), \mu^{\setminus i}(t)) \qquad \rho^i(T) = \text{given},$$
$$\frac{d}{dt}\mu^i(t) = \beta(\mu^i(t), \rho^i(t), \mu^{\setminus i}(t)) \quad \mu^i(0) = \text{given}.$$

where α and β are defined by the right-hand side of the differential equations (16). Since the evolution of ρ^i does not depend on μ^i, we can integrate backward from time T to solve for ρ^i. Then, integrating forward from time 0, we compute μ^i. After performing a single iteration of backward-forward integration, we obtain a solution that satisfies the fixed-point equation (16) for the i'th component (this is not surprising once we have identified our procedure to be a variation of a standard forward-backward algorithm for a single component). Such a solution will be a local maximum of the functional w.r.t. to η^i (reaching a local minimum or a saddle point requires very specific initialization points).

This suggests that we can use the standard procedure of asynchronous updates, where we update each component in a round-robin fashion. Since each of these single-component updates converges in one backward-forward step, and since it reaches a local maximum, each step improves the value of the free energy over the previous one. As the free energy functional is bounded by the probability of the evidence, this procedure will always converge, and the rate of the free energy increase can be used to test for convergence.

Potentially, there can be many scheduling possibilities. In our implementation the update scheduling is simply random. A better choice would be to update the component which would maximally increase the value of the functional in that iteration. This idea is similar to the scheduling of Elidan et al. (2006), who approximate the change in the beliefs by bounding the *residuals* of the messages, which give an approximation of the benefit of updating each component.

Another issue is the initialization of this procedure. Since the iteration on the i'th component depends on $\mu^{\setminus i}$, we need to initialize μ by some legal assignment. To do so, we create a fictional rate matrix $\tilde{\mathbb{Q}}_i$ for each component and initialize μ^i to be the posterior of the process given the evidence $e_{i,0}$ and $e_{i,T}$. As a reasonable initial guess, we choose at random one of the conditional rates $\mathbb{Q}^{i|u_i}$ using some random assignment u_i to determine the fictional rate matrix.

The general optimization procedure is summarized in the following algorithm:

For each i, initialize μ^i using some legal marginal function.
while *not converged* **do**

1. *Pick a component $i \in \{1,\ldots,D\}$.*
2. *Update $\rho^i(t)$ by solving the ρ^i backward differential equation in (16).*
3. *Update $\mu^i(t)$ and $\gamma^i(t)$ by solving the μ^i forward differential equation in (16) and using the algebraic equation in (17).*

end

Algorithm 1: Mean field approximation in continuous-time Bayesian networks

4.3 Exploiting Continuous-Time Representation

The continuous-time update equations allow us to use standard ODE methods with an adaptive step size (here we use the Runge-Kutta-Fehlberg (4,5) method). At the price of some overhead, these procedures automatically tune the trade-off between error and time granularity. Moreover, this overhead is usually negligible compared to the saving in computation time, because adaptive integration can be more efficient than *any* fixed step size integration by an order of magnitude (Press et al., 2007).

To further save computations, we note that while standard integration methods involve only initial boundary conditions at $t = 0$, the solution of μ^i is also known at $t = T$. Therefore, we stop the adaptive integration when $\mu^i(t) \approx \mu^i(T)$ and t is close enough to T. This modification reduces the number of computed points significantly because the derivative of μ^i tends to grow near the boundary, resulting in a smaller step size.

The adaptive solver selects different time points for the evaluation of each component. Therefore, updates of η^i require access to marginal density sets of neighboring components at time points that differ from their evaluation points. To allow efficient interpolation, we use a piecewise linear approximation of η whose boundary points are determined by the evaluation points that are chosen by the adaptive integrator.

5. Perspectives and Related Work

Variational approximations for different types of continuous-time processes have been recently proposed. Examples include systems with discrete hidden components (Opper and Sanguinetti, 2007); continuous-state processes (Archambeau et al., 2007); hybrid models involving both discrete and continuous-time components (Sanguinetti et al., 2009; Opper and Sanguinetti, 2010); and spatiotemporal processes (Ruttor and Opper, 2010; Dewar et al., 2010). All these models assume noisy observations in a finite number of time points. In this work we focus on structured discrete-state processes with noiseless evidence.

Our approach is motivated by results of Opper and Sanguinetti (2007) who developed a variational principle for a related model. Their model is similar to an HMM, in which the hidden chain is a continuous-time Markov process and there are (noisy) observations at discrete points along the process. They describe a variational principle and discuss the form of the functional when the approximation is a product of independent processes. There are two main differences between the setting of Opper and Sanguinetti and ours. First, we show how to exploit the structure of the target CTBN to reduce the complexity of the approximation. These simplifications imply that the update of the i'th process depends only on its Markov blanket in the CTBN, allowing us to develop efficient approximations for large models. Second, and more importantly, the structure of the evidence in our setting is quite different, as we assume deterministic evidence at the end of intervals. This setting typically leads to a posterior Markov process in which the instantaneous rates used by Opper and Sanguinetti diverge toward the end point—the rates of transition into the observed state go to infinity, leading to numerical problems at the end points. We circumvent this problem by using the marginal density representation which is much more stable numerically.

Taking the general perspective of Wainwright and Jordan (2008), the representation of the distribution uses the natural sufficient statistics. In the case of a continuous-time Markov process, the sufficient statistics are T_x, the time spent in state x, and $M_{x,y}$, the number of transitions from state x to y. In a discrete-time model, we can capture the statistics for every random variable. In a continuous-time model, however, we need to consider the time derivative of the statistics. Indeed, as shown in Section 3.3 we have

$$\frac{d}{dt}\mathbf{E}\left[T_x(t)\right] = \mu_x(t) \quad \text{and} \quad \frac{d}{dt}\mathbf{E}\left[M_{x,y}(t)\right] = \gamma_{x,y}(t).$$

Thus, our marginal density sets η provide what we consider a natural formulation for variational approaches to continuous-time Markov processes.

Our presentation focused on evidence at two ends of an interval. Our formulation easily extends to deal with more elaborate types of evidence: (1) If we do not observe the initial state of the i'th component, we can set $\mu^i_x(0)$ to be the prior probability of $X^{(0)} = x$. Similarly, if we do not observe X_i at time T, we set $\rho^i_x(T) = 1$ as initial data for the backward step. (2) In a CTBN where one (or more) components are fully observed throughout some interval, we simply set μ^i for these components to be a distribution that assigns all the probability mass to the observed trajectory. Similarly, if we observe different components at different times, we may update each component on a different time interval. Consequently, maintaining for each component a marginal distribution μ^i throughout the interval of interest, we can update the other ones using their evidence patterns.

6. Evaluation on Ising Chains

To gain better insight into the quality of our procedure, we performed numerical tests on models that challenge the approximation. Specifically, we use Ising chains with the parameterization introduced in Example 1, where we explore regimes defined by the degree of coupling between the components (the parameter β) and the rate of transitions (the parameter τ). We evaluate the error in two ways. The first is by the difference between the true log-likelihood and our estimate. The second is by the average relative error in the estimate of different expected sufficient statistics defined by $\sum_j |\hat{\theta}_j - \theta_j|/\theta_j$, where θ_j is exact value of the j'th expected sufficient statistics and $\hat{\theta}_j$ is the approximation. To obtain a stable estimate the average is taken over all $\theta_j > 0.05 \max_{j'} \theta_{j'}$.

Applying our procedure on an Ising chain with 8 components, for which we can still perform exact inference, we evaluated the relative error for different choices of β and τ. The evidence in this experiment is $e_0 = \{+,+,+,+,+,+,-,-\}$, $T = 0.64$ and $e_T = \{-,-,-,+,+,+,+,+\}$. As shown in Figure 4(a), the error is larger when τ and β are large. In the case of a weak coupling (small β), the posterior is almost factored, and our approximation is accurate. In models with few transitions (small τ), most of the mass of the posterior is concentrated on a few canonical "types" of trajectories that can be captured by the approximation (as in Example 3). At high transition rates, the components tend to transition often, and in a coordinated manner, which leads to a posterior that is hard to approximate by a product distribution. Moreover, the resulting free energy landscape is rough with many local maxima. Examining the error in likelihood estimates (Figure 4(b),(c)) we see a similar trend.

Next, we examine the run time of our approximation when using fairly standard ODE solver with few optimizations and tunings. The run time is dominated by the time needed to perform the backward-forward integration when updating a single component, and by the number of such updates until convergence. Examining the run time for different choices of β and τ (Figure 5), we see that the run time of our procedure scales linearly with the number of components in the chain. The differences among the different curves suggest that the runtime is affected by the choice of parameters, which in turn affect the smoothness of the posterior density sets.

7. Evaluation on Branching Processes

The above-mentioned experimental results indicate that our approximation is accurate when reasoning about weakly-coupled components, or about time intervals involving few transitions (low transition rates). Unfortunately, in many domains we face strongly-coupled components. For example, we are interested in modeling the evolution of biological sequences (DNA, RNA, and proteins).

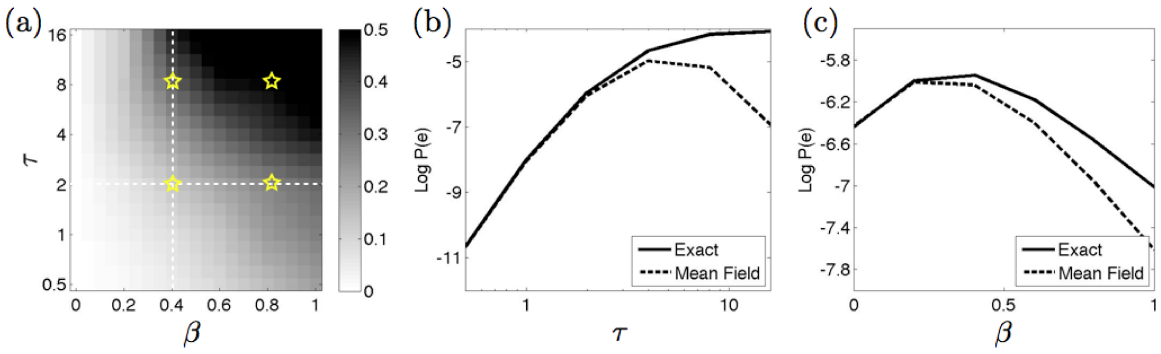


Figure 4: *(a)* Relative error as a function of the coupling parameter β (*x*-axis) and transition rates τ (*y*-axis) for an 8-component Ising chain. *(b)* Comparison of true vs. estimated likelihood as a function of the rate parameter τ. *(c)* Comparison of true vs. likelihood as a function of the coupling parameter β.

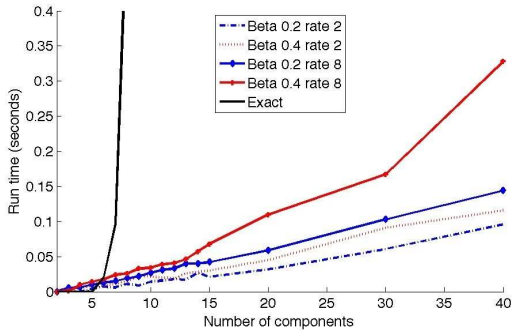


Figure 5: Evaluation of the run time of the approximation versus the run time of exact inference as a function of the number of components.

In such systems, we have a *phylogenetic tree* that represents the branching process that leads to current day sequences (see Figure 6).

It is common in sequence evolution to model this process as a continuous-time Markov process over a tree (Felsenstein, 2004). More precisely, the evolution along each branch is a standard continuous-time Markov process, and branching is modeled by a replication, after which each replica evolves independently along its sub-branch. Common applications are forced to assume that each character in the sequence evolves independently of the other.

In some situations, assuming an independent evolution of each character is highly unreasonable. Consider the evolution of an RNA sequence that folds onto itself to form a functional structure, as in Figure 7(a). This folding is mediated by complementary base-pairing (A-U, C-G, etc) that stabilizes the structure. During evolution, we expect to see compensatory mutations. That is, if a *A* changes into *C* then its based-paired *U* will change into a *G* soon thereafter. To capture such

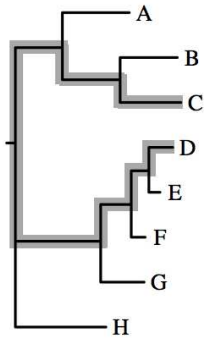


Figure 6: An example of a phylogenetic tree. Branch lengths denote time intervals between events. The interval used for the comparison with non-branching processes is highlighted.

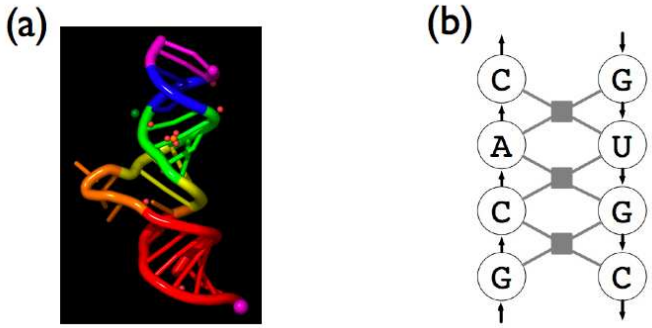


Figure 7: *(a)* Structure of an RNA molecule. The 3 dimensional structure dictates the dependencies between the different positions. *(b)* The form of the energy function for encoding RNA folding, superimposed on a fragment of a folded structure; each gray box denotes a term that involves four nucleotides.

coordinated changes, we need to consider the joint evolution of the different characters. In the case of RNA structure, the stability of the structure is determined by *stacking potentials* that measure the stability of two adjacent pairs of interacting nucleotides. Thus, if we consider a factor network to represent the energy of a fold, it will have structure as shown in Figure 7(b). We can convert this factor graph into a CTBN using procedures that consider the energy function as a fitness criteria in evolution (El-Hay et al., 2006; Yu and Thorne, 2006). Unfortunately, inference in such models suffers from computational blowup, and so the few studies that deal with it explicitly resort to sampling procedures (Yu and Thorne, 2006).

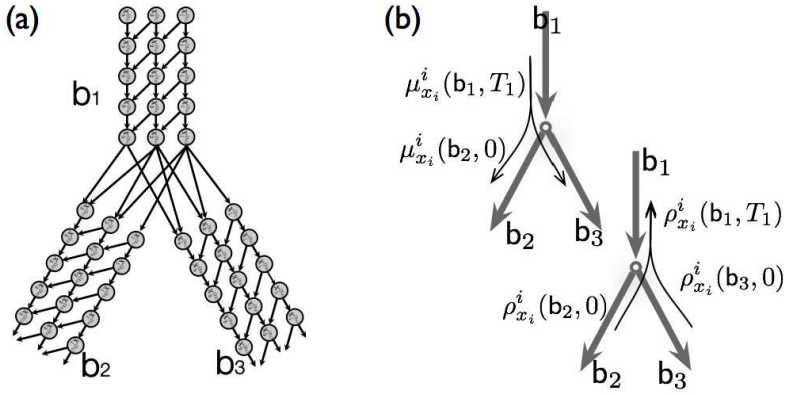


Figure 8: Structure of the branching process. *(a)* The discretized CTBN underlying the process in an intersection. *(b)* Illustration of the ODE updates on a directed tree, updating $\rho^i(t)$ backwards using (21) and $\mu^i(t)$ forwards using (22).

7.1 Representation

To consider phylogenetic trees, we should take a common approach in evolutionary analysis, in which inference of the tree topology and branch lengths is performed separately from inference of sequence dynamics. Thus, we need to extend our framework to deal with branching processes, where the branching points are fixed and known. In a linear-time model, we view the process as a map from $[0,T]$ into random variables $X^{(t)}$. In the case of a tree, we view the process as a map from a point $\mathsf{t} = \langle \mathsf{b}, t \rangle$ on a tree $\mathcal{T}$ (defined by branch b and the time t within it) into a random variable $X^{(\mathsf{t})}$. Similarly, we generalize the definition of the Markov-consistent density set η to include functions on trees. We define continuity of functions on trees in the obvious manner.

To gain intuition on this process we return to the discrete case, where our branching process can be viewed as a branching of the Dynamic Bayesian Network from one branch to two separate branches at the vertex, as in Figure 8(a).

7.2 Inference on Trees

The variational approximation on trees is thus similar to the one on intervals. Within each branch, we deal with the same update formulas as in linear time. We denote by $\mu^i_{x_i}(\mathsf{b},t)$ and $\rho^i_{x_i}(\mathsf{b},t)$ the messages computed on branch b at time t. The only changes occur at vertices, where we cannot use the Euler-Lagrange equations (Appendix E), therefore we must derive the propagation equations using a different method.

The following proposition establishes the update equations for the parameters $\mu^i(t)$ and $\rho^i(t)$ at the vertices, as depicted in Figure 8(b):

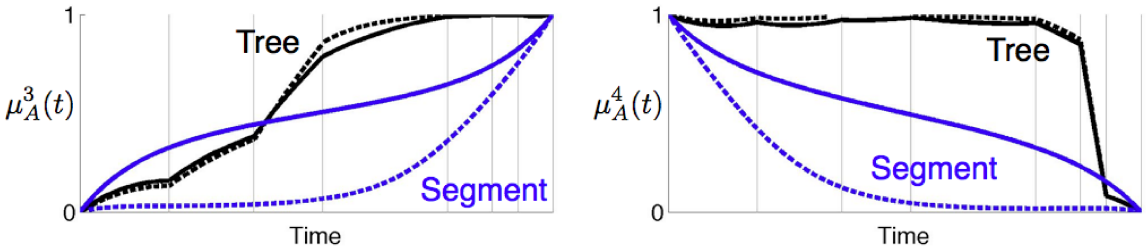


Figure 9: Comparison of exact vs. approximate inference along the highlighted path from C to D in the tree of Figure 6 with and without additional evidence at other leaves. In the latter case the problem is equivalent to inference on a linear segment. Exact marginals are shown in solid lines, whereas approximate marginals are in dashed lines. The horizontal gray lines indicate branch points along the path. Notice that evidence at the leaves result in discontinuities of the derivatives at such points. The two panels show two different components.

Proposition 7 *Given a vertex T with an incoming branch b_1 and two outgoing branches $\mathsf{b}_2, \mathsf{b}_3$. The following are the correct updates for our parameters $\mu^i_{x_i}(t)$ and $\rho^i_{x_i}(t)$:*

$$\rho^i_{x_i}(\mathsf{b}_1, T) = \rho^i_{x_i}(\mathsf{b}_2, 0)\rho^i_{x_i}(\mathsf{b}_3, 0), \tag{21}$$

$$\mu^i_{x_i}(\mathsf{b}_k, 0) = \mu^i_{x_i}(\mathsf{b}_1, T) \qquad k = 2, 3. \tag{22}$$

Proof See Appendix G ■

Using Proposition 7 we can set the updates of the different parameters in the branching process according to (21–22). In the backward propagation of ρ^i, the value at the end of b_1 is the product of the values at the start of the two outgoing branches. This is the natural operation when we recall the interpretation of ρ^i as the probability of the downstream evidence given the current state (which is its exact meaning in a single component process): the downstream evidence of b_2 is independent of the downstream evidence of b_3, given the state of the process at the vertex $\langle \mathsf{b}_1, T \rangle$. The forward propagation of μ^i simply uses the value at the end of the incoming branch as initial value for the outgoing branches.

When switching to trees, we essentially increase the amount of evidence about intermediate states. Consider for example the tree of Figure 6 with an Ising chain model (as in the previous subsection). We can view the span from C to D as an interval with evidence at its end. When we add evidence at the tip of other branches we gain more information about intermediate points between C and D. Even though this evidence can represent evolution from these intermediate points, they do change our information state about them. To evaluate the impact of these changes on our approximation, we considered the tree of Figure 6, and compared it to inference in the backbone between C and D (Figure 4). Comparing the true marginal to the approximate one along the main backbone (see Figure 9) we see a major difference in the quality of the approximation. The evidence in the tree leads to a much tighter approximation of the marginal distribution. A more systematic

comparison (Figure 10) demonstrates that the additional evidence reduces the magnitude of the error throughout the parameter space.

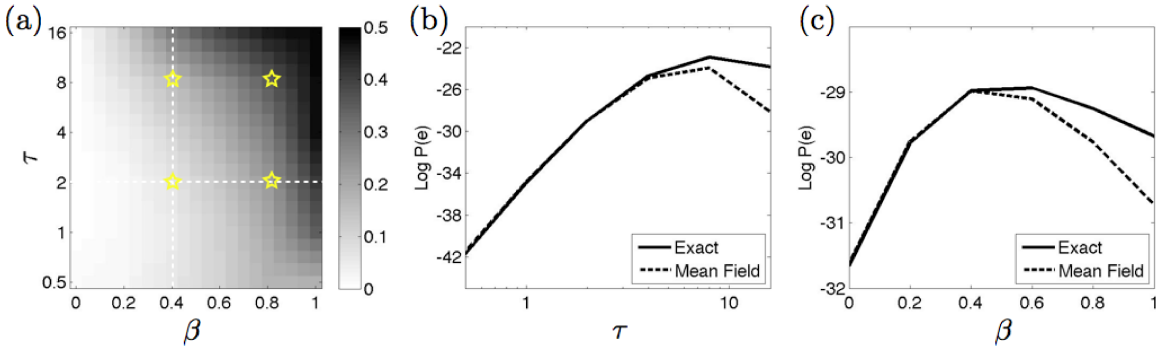


Figure 10: *(a)* Evaluation of the relative error in expected sufficient statistics for an Ising chain in branching-time; compare to Figure 4(a). *(b),(c)* Evaluation of the estimated likelihood on a tree w.r.t. the rate τ and coupling β; compare to Figure 4(b),(c).

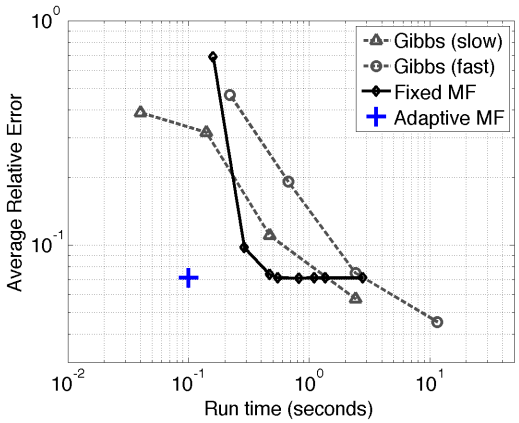


Figure 11: Evaluation of the run time vs. accuracy trade-off for several choices of parameters for mean field and Gibbs sampling on the branching process of Figure 6.

Similarly to mean-field, the Gibbs sampling procedure for CTBNs (El-Hay et al., 2008) can also be extended to deal with branching processes. Comparing our method to the Gibbs sampling procedure we see (Figure 11) that the faster mean field approach dominates the Gibbs procedure over short run times. However, as opposed to mean field, the Gibbs procedure is asymptotically unbiased, and with longer run times it ultimately prevails. This evaluation also shows that the adaptive integration procedure in our methods strikes a better trade-off than using a fixed time granularity integration.

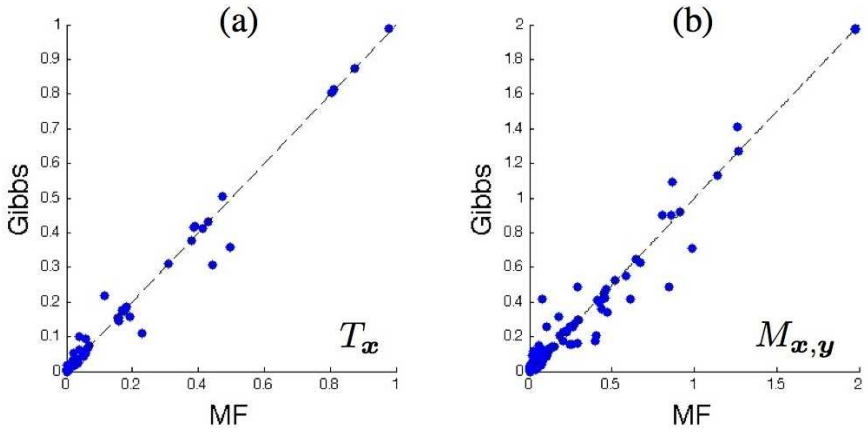


Figure 12: Comparison of estimates of expected sufficient statistics in the evolution of 18 interacting nucleotides, using a realistic model of RNA evolution. Each point is an expected value of: (a) residence time in a specific state of a component and its parents; (b) number of transition between two states. The x-axis is the estimate by the variational procedure, whereas the y-axis is the estimate by Gibbs sampling.

As a more demanding test, we applied our inference procedure to a model similar to the one introduced by Yu and Thorne (2006) for a stem of 18 interacting RNA nucleotides in 8 species in the phylogeny of Figure 6. In this model the transition rate between two sequences that differ in a single nucleotide depends on difference between their folding energy. Specifically, the transition rate from sequence x to sequence y is given by

$$q_{x,y} = 1.6\left(1 + e^{E_{\text{fold}}(y) - E_{\text{fold}}(x)}\right)^{-1}, \qquad |\delta(x,y)| = 1,$$

where E_{fold} is the folding energy of the sequence. This equation implies that transition rates are increasing monotonically with the reduction of the folding energy. Hence, this model tends to evolve into low energy states. The folding energy in turn is a sum of local stacking energies, involving quadruples of nucleotides as described by the factors in Figure 7. Denoting the subset of positions contained in each quadruple by D_k, the energy is

$$E_{\text{fold}}(x) = \sum_k E^k_{\text{fold}}(x|_{D_k}),$$

where $x|_{D_k}$ is the subset of nucleotides that belong factor k. This model is equivalent to a CTBN in which the parents of each components are the other components that share the same factors. This property follows from the fact that for any pair x and y, where $\delta(x,y) = \{i\}$, the difference between the energies of these two sequences depends only on the factors that contain i.

We compared our estimate of the expected sufficient statistics of this model to these obtained by the Gibbs sampling procedure. The Gibbs sampling estimates were chosen by running the procedure with an increasing computation time until there was no significant change in the results. The

final estimates was obtained using 5000 burn-in rounds, 10000 number of samples and 100 rounds between two consecutive samples. The results, shown in Figure 12, demonstrate that over all the two approximate inference procedures are in good agreement about the value of the expected sufficient statistics.

8. Discussion

In this paper we formulate a general variational principle for continuous-time Markov processes (by reformulating and extending the one proposed by Opper and Sanguinetti, 2007), and use it to derive an efficient procedure for inference in CTBNs. In this mean field approximation, we use a product of independent inhomogeneous processes to approximate the multi-component posterior.

Our procedure enjoys the same benefits encountered in discrete-time mean field procedure (Jordan et al., 1999): it provides a lower-bound on the likelihood of the evidence and its run time scales linearly with the number of components. Using asynchronous updates it is guaranteed to converge, and the approximation represents a consistent joint distribution. It also suffers from expected shortcomings: the functional has multiple local maxima, it cannot capture complex interactions in the posterior (Example 3). By using a time-inhomogeneous representation our approximation does capture complex patterns in the temporal progression of the marginal distribution of each component. Importantly, the continuous-time parameterization enables straightforward implementation using standard ODE integration packages that automatically tune the trade-off between time granularity and approximation quality. We show how it is extended to perform inference on phylogenetic trees, where the posterior distribution is directly affected by several evidence points, and show that it provides fairly accurate answers in the context of a real application (Figure 12).

A key development is the introduction of marginal density sets. Using this representation we reformulate and extend the variational principle proposed by Opper and Sanguinetti (2007) , which incorporates a different inhomogeneous representation. This modification allows handling direct evidence of the state of the process, as in the case of CTBNs, while keeping the representation of the approximation bounded. The extension of this principle to CTBNs follows by exploiting their networks structure. This adaptation of continuously inhomogeneous representations to CTBNs increases the flexibility of the approximation relative to the piecewise homogeneous representation of Saria et al. (2007) and, somewhat surprisingly, also significantly simplifies the resulting formulation.

The proposed representation is natural in the sense that its functions are the time-derivatives of the expected sufficient statistics that we are willing to evaluate. Hence, once finding the optimal value of the lower bound, calculating these expectations is straightforward. This representation is analogous to mean parameters which have proved powerful in variational approximations of exponential families over finite random variable sets (Wainwright and Jordan, 2008).

We believe that even in cases where evidence is indirect and noisy, the marginal density representation should comprise smoother functions than posterior rates. Intuitively, in the presence of a noisy observation the posterior probability of some state x can be very small. In such cases, the posterior transition rate form x into a state that better explains the observation might tend to a large quantity. This reasoning suggests that marginal density representations should be better handled by adaptive numerical integration algorithms. An interesting direction would be to test this conjecture empirically.

A possible extension is using our variational procedure to generate the initial distribution for the Gibbs sampling procedure and thus skip the initial burn-in phase and produce accurate samples. Another attractive aspect of this new variational approximation is its potential use for learning model parameters from data. It can be easily combined with the EM procedure for CTBNs (Nodelman et al., 2005a) to obtain a Variational-EM procedure for CTBNs, which monotonically increases the likelihood by alternating between steps that improve the approximation η (the updates discussed here) and steps that improve the model parameters θ (an M-step Nodelman et al., 2005a). Finally, marginal density sets are a particularly suitable representation for adapting richer representations such as Bethe, Kikuchi and convex approximations to non-homogeneous versions (El-Hay et al., 2010). Further work in that direction should allow bridging the gap in the wealth of inference techniques between finite domain models and continuous-time models.

Acknowledgments

We thank Ofer Meshi, Menachem Fromer and the anonymous reviewers for helpful remarks on previous versions of the manuscript. This research was supported in part by a grant from the Israel Science Foundation. Tal El-Hay is supported by the Eshkol fellowship from the Israeli Ministry of Science.

Appendix A. The Relation Between CTBNs and DBNs

In this section we show that the DBN construction of Equations (6-7) is such that as h approaches 0, the distribution P_h approaches Pr. To show this, it suffice to show that

$$\lim_{h\to 0} \frac{P_h(X^{(t_{k+1})} = y | X^{(t_k)} = x) - \mathbf{1}_{x=y}}{h} = q_{x,y} \ .$$

We ensured this condition holds component-wise, and now need to show that this leads to global consistency.

Plugging Equation (7) into Equation (6), the transition probability of the DBN is

$$P_h(X^{(t_{k+1})} = y | X^{(t_k)} = x) = \prod_i \left(\mathbf{1}_{x_i=y_i} + q^{i|\mathbf{Pa}_i}_{x_i,y_i|u_i} \cdot h \right) \ .$$

Since we consider the limit as h approaches 0, any term that involves h^d with $d > 1$ is irrelevant. And thus, we can limit our attention to the constant terms and terms linear in h. Expanding the product gives

$$P_h(X^{(t_{k+1})} = y | X^{(t_k)} = x) = \prod_i \mathbf{1}_{x_i=y_i} + \sum_i q^{i|\mathbf{Pa}_i}_{x_i,y_i|u_i} \cdot h \prod_{j\neq i} \mathbf{1}_{x_j=y_j} + o(h) \ .$$

Now, $\prod_i \mathbf{1}_{x_i=y_i} = \mathbf{1}_{x=y}$. Moreover, it is easy to verify that

$$q_{x,y} = \sum_i q^{i|\mathbf{Pa}_i}_{x_i,y_i|u_i} \prod_{j\neq i} \mathbf{1}_{x_j=y_j} \ .$$

Thus,

$$P_h(X^{(t_{k+1})} = y | X^{(t_k)} = x) = \mathbf{1}_{x=y} + q_{x,y} h + o(h),$$

proving the required condition.

Appendix B. Marginal Density Sets and Markov Processes - Proof of Lemma 2

Proof Given η, we define the *inhomogeneous rate matrix* $\mathbb{R}(t)$ as in Equation (11). $\mathbb{R}(t)$ is a valid rate matrix because its off-diagonals are non-negative as they are the quotient of two non-negative functions, and because applying the requirement on $\gamma_{x,x}(t)$ in Definition 1

$$r_{x,x}(t) = \frac{\gamma_{x,x}(t)}{\mu_x(t)} = -\frac{\sum_{y \neq x} \gamma_{x,y}(t)}{\mu_x(t)} = -\sum_{y \neq x} r_{x,y}(t) \ ,$$

we see that $\mathbb{R}(t)$'s diagonals are negative and the rows sum to 0. We can use these rates with the initial value $\mu_x(0)$ to construct the Markov process P_η from the forward master equation

$$\frac{d}{dt} P_\eta(X^{(t)} = x) = \sum_y P_\eta(X^{(t)} = y) r_{y,x}(t) \ ,$$

and

$$P_\eta(X^{(0)}) = \mu(0) \ .$$

To conclude the proof we show that P_η and the marginal density set satisfy (10). First, from Definition 1 it follows that $\mu(t)$ is the solution to the master equation of $P_\eta(X^{(t)})$, because the initial values match at $t = 0$ and the time-derivatives of the two functions are identical. Thus

$$P_\eta(X^{(t)} = x) = \mu_x(t) \ .$$

Next, the equivalence of the joint probability densities can be proved:

$$\begin{aligned} \lim_{h \to 0} \frac{\Pr(X^{(t)} = x, X^{(t+h)} = y)}{h} &= \lim_{h \to 0} \frac{\mu_x(t) \Pr(X^{(t+h)} = y | \Pr(X^{(t)} = x)}{h} \\ &= \lim_{h \to 0} \frac{\mu_x(t) r_{x,y}(t) h}{h} \\ &= \mu_x(t) r_{x,y}(t) \ . \end{aligned}$$

From the definition of $r_{x,y}(t)$ and the fact that $\gamma_{x,y}(t) = 0$ whenever $\mu_x(t) = 0$, it follows that $\mu_x(t) r_{x,y}(t)$ is exactly $\gamma_{x,y}(t)$ ■

Appendix C. Expectations in Inhomogeneous Processes

This section includes the proofs of the lemmas used in the proof of the variational lower bound theorem.

C.1 Expectations of Functions of States - Proof of Lemma 4

Proof Changing the order of integration we obtain

$$\mathbf{E}_{f_\eta} \left[\int_0^T \psi(\sigma(t), t) dt \right] \equiv \int_\Sigma f_\eta(\sigma) \int_0^T \psi(\sigma(t), t) dt d\sigma = \int_0^T \int_\Sigma f_\eta(\sigma) \cdot \psi(\sigma(t), t) \, d\sigma dt \ .$$

For each $t \in T$ we decompose the inner integral according to possible states at that time:

$$\begin{aligned}\int_\Sigma f_\eta(\sigma)\cdot\psi(\sigma(t),t)\,d\sigma &= \sum_x \int_\Sigma f_\eta(\sigma)\cdot \mathbf{1}_{\sigma(t)=x}\cdot\psi(x,t)\,d\sigma \\ &= \sum_x \psi(x,t)\int_\Sigma f_\eta(\sigma)\cdot \mathbf{1}_{\sigma(t)=x}\,d\sigma \\ &= \sum_x \psi(x,t)\mu_x(t)\,.\end{aligned}$$

■

C.2 Expectations of Functions of Transitions - Proof of Lemma 5

Proof Given a trajectory σ there exists a small enough $h>0$ such that for every transition and for every $t \in (t_k-h,t_k)$ we have $\sigma(t)=x_{k-1}$ and $\sigma(t+h)=x_k$. In that case we can rewrite the sum in the expectation term as

$$\begin{aligned}\sum_{k=1}^{K^\sigma}\psi(x^\sigma_{k-1},x^\sigma_k,t^\sigma_k) &= \sum_{k=1}^{K^\sigma}\frac{1}{h}\int_{t_k-h}^{t_k}\psi(\sigma(t),\sigma(t+h),t)dt+\frac{o(h)}{h} \\ &= \frac{1}{h}\int_0^{T-h}\psi(\sigma(t),\sigma(t+h),t)dt+\frac{o(h)}{h}\,,\end{aligned}$$

where the first equality follows from continuity and the second one from the requirement that $\psi(x,x,t)=0$. Taking the limit $h\to 0$ and using this requirement again gives

$$\sum_{k=1}^{K^\sigma}\psi(x^\sigma_{k-1},x^\sigma_k,t^\sigma_k)=\frac{d}{ds}\left[\int_0^T\psi(\sigma(t),\sigma(t+s),t)dt\right]_{s=0}\,.$$

Taking expectation we obtain

$$\begin{aligned}&\int_\Sigma f(\sigma)\frac{d}{ds}\left[\int_0^T\psi(\sigma(t),\sigma(t+s),t)dt\right]_{s=0}d\sigma \\ &= \int_\Sigma f(\sigma)\frac{d}{ds}\left[\int_0^T\sum_x\sum_{y\neq x}\psi(x,y,t)\,\mathbf{1}_{\sigma(t)=x}\mathbf{1}_{\sigma(t+s)=y}\,dt\right]_{s=0}d\sigma \\ &= \frac{d}{ds}\left[\int_0^T\sum_x\sum_{y\neq x}\psi(x,y,t)\int_\Sigma f(\sigma)\,\mathbf{1}_{\sigma(t)=x}\mathbf{1}_{\sigma(t+s)=y}d\sigma\,dt\right]_{s=0}\,.\end{aligned}$$

The inner integral in the last term is a joint probability

$$\int_\Sigma f(\sigma)\mathbf{1}_{\sigma(t)=x}\mathbf{1}_{\sigma(t+s)=y}d\sigma=\Pr(X^{(t)}=x,X^{(t+s)}=y)\,.$$

Switching the order of integration and differentiation and using

$$\frac{d}{ds}\Pr(X^{(t)}=x,X^{(t+s)}=y)\Big|_{s=0}=\gamma_{xy}(t),\quad x\neq y,$$

gives the desired result. ■

Appendix D. Proof of the Factored Representation of the Energy Functional

Proof We begin with the definition of the average energy

$$\begin{aligned}\mathcal{E}(\eta;\mathbb{Q}) &= \int_0^T \sum_x \left[\mu_x(t)q_{x,x} + \sum_{y\neq x}\gamma_{x,y}(t)\ln q_{x,y}\right]dt \\ &= \int_0^T \sum_x \left[\mu_x(t)q_{x,x} + \sum_i \sum_{y_i\neq x_i}\gamma^i_{x_i,y_i}(t)\mu^{\setminus i}(t)\ln q_{x,y}\right]dt\ .\end{aligned}$$

where the equality stems from the observation that the only states y that may have $\gamma_{x,y}(t) > 0$, are those with $\delta(x,y) \leq 1$ (all the rest are 0). Thus, the enumeration over all possible states collapses into an enumeration over all components i and all states $y_i \neq x_i$. Due to the fact that we are only considering transitions in single components, we may replace the global joint density $\gamma_{x,y}$ with $\gamma^i_{x_i,y_i}\cdot\mu^{\setminus i}(t)$, as per definition.

Using (5), we can decompose the transition rates $q_{x,x}$ and $q_{x,y}$ to get

$$\begin{aligned}\mathcal{E}(\eta;\mathbb{Q}) &= \sum_i \int_0^T \sum_x \left[\mu_x(t)q_{x_i,x_i|u_i} + \sum_{y_i\neq x_i}\gamma^i_{x_i,y_i}(t)\mu^{\setminus i}(t)\ln q_{x_i,y_i|u_i}\right]dt \\ &= \sum_i \int_0^T \sum_{x_i} \left[\mu^i_{x_i}(t)\sum_{x\setminus i}\mu^{\setminus i}_{x\setminus i}(t)q_{x_i,x_i|u_i} + \sum_{y_i\neq x_i}\gamma^i_{x_i,y_i}(t)\mu^{\setminus i}_{x\setminus i}(t)\ln q_{x_i,y_i|u_i}\right]dt\ .\end{aligned}$$

To get to the last equality we use the factorization of $\mu(t)$ as a product of $\mu^i(t)$ with $\mu^{\setminus i}(t)$ and the reordering of the summation. Next we simply write the previous sum as an expectation over $X\setminus i$

$$\mathcal{E}(\eta;\mathbb{Q}) = \sum_i \int_0^T \sum_{x_i}\mu^i_{x_i}(t)\mathbf{E}_{\mu^{\setminus i}(t)}\left[q_{x_i,x_i|U_i}\right] + \sum_i \int_0^T \sum_{y_i\neq x_i}\gamma^i_{x_i,y_i}(t)\mathbf{E}_{\mu^{\setminus i}(t)}\left[\ln q_{x_i,y_i|U_i}\right]dt\ ,$$

which concludes the proof.

Turning to the entropy-like term we have:

$$\begin{aligned}\mathcal{H}(\eta) &= \int_0^T \sum_x\sum_{y\neq x}\gamma_{x,y}(t)[1+\ln\mu_x(t)-\ln\gamma_{x,y}(t)]dt \\ &= \sum_i\int_0^T \sum_x\sum_{y_i\neq x_i}\mu^{\setminus i}(t)\gamma_{x_i,y_i}(t)[1+\sum_i\ln\mu^i_{x_i}(t)-\ln\gamma_{x_i,y_i}(t)-\sum_{j\neq i}\ln\mu_{x_j}(t)]dt \\ &= \sum_i\int_0^T \sum_{x_i}\sum_{y_i\neq x_i}\gamma_{x_i,y_i}(t)[1+\ln\mu^i_{x_i}(t)-\ln\gamma_{x_i,y_i}(t)]dt \\ &= \sum_i \mathcal{H}(\eta^i)\ ,\end{aligned}$$

where, the first equality is definition of $\mathcal{H}$. The second one follows from the definition of the factored density set. The third one is obtained by algebraic manipulation and the last one is again the definition of $\mathcal{H}$. ■

Appendix E. Euler-Lagrange Equations

The problem of finding the fixed points of *functionals* whose arguments are continuous functions comes from the field of *Calculus of variations*. We briefly review the usage Euler-Lagrange equation for solving optimization problems over functionals. Additional information can be found in Gelfand and Fomin (1963).

A functional is a mapping from a vector space to its underlying field. In our case the functional is the Lagrangian introduced in Section 4, which is an integral over real-valued functions, and the underlying field is the real numbers.

Given a functional over a normed space of continuously differentiable real functions of the form

$$I[y] = \int_a^b f(t, y(t), y'(t))dt$$

where $y'(t)$ is the time-derivative of the function $y(t)$, we would like to find a function $y(t)$ that minimizes (or in our case maximizes) the functional subject to $y(a) = y_a$ and $y(b) = y_b$. In the simplest case, when there are no additional constraints, a necessary condition for y to be a local optimum is that y is a *stationary point*. Roughly, a stationary point is a function y, where $I[y]$ is insensitive to small variations in y. That is, given a function $h(t)$ where $h(a) = 0$ and $h(b) = 0$, the change of the functional $I[y+h] - I[y]$ is small relative to the norm of h. For $y(t)$ to be a stationary point, it must satisfy the *Euler-Lagrange* equations (Gelfand and Fomin, 1963)

$$\frac{\partial}{\partial y} f(t, y(t), y'(t)) - \frac{d}{dt}\left(\frac{\partial}{\partial y'} f(t, y(t), y'(t))\right) = 0 \ . \tag{23}$$

In this paper we have additional constraints describing the time derivative of μ. The generalization of the Euler-Lagrange equations to that case is straightforward. Denoting the subsidiary constraints by $g(t, y(t), y'(t)) = 0$, we simply replace $f(t, y, y')$ by $f(t, y, y') - \lambda(t)g(t, y, y')$ in Equation 23.

An example for the use of this equation is in the following proof.

Appendix F. Stationary Points of the Lagrangian - Proof of Theorem 6

Proof For convenience, we begin by rewriting the Lagrangian in explicit form: $\mathcal{L} = \int_0^T f(y(t), y'(t))dt$ where $y(t) = \langle \mu(t), \gamma(t), \lambda(t) \rangle$ is a concatenation of the parameters and Lagrange multiplier and

$$\begin{aligned} f(y(t), y'(t)) \quad = \quad & \sum_{i=1}^{D} \sum_{x_i} \Bigg[\mu^i_{x_i}(t) \mathbf{E}_{\mu^{\setminus i}(t)} \left[q_{x_i, x_i | U_i} \right] + \sum_{y_i \neq x_i} \gamma^i_{x_i, y_i}(t) \mathbf{E}_{\mu^{\setminus i}(t)} \left[\ln q_{x_i, y_i | U_i} \right] \\ & + \sum_{y_i \neq x_i} \gamma_{x_i y_i} \left[1 + \ln \mu^i_{x_i}(t) - \ln \gamma^i_{x_i y_i}(t) \right] - \lambda^i_{x_i}(t) \left(\frac{d}{dt} \mu^i_{x_i}(t) - \sum_{y_i} \gamma^i_{x_i y_i}(t) \right) \Bigg] \ . \end{aligned}$$

The Euler-Lagrange equations of the Lagrangian define its stationary points w.r.t. the parameters of each component $\mu^i(t), \gamma^i(t)$ and $\lambda^i(t)$.

First, we take the partial derivatives of f w.r.t to $\mu^i_{x_i}(t)$ as well as $\frac{d}{dt}\mu^i_{x_i}(t)$ and plug them into Equation 23. We start by handling the energy terms. These terms involve expectations in the form

$\mathbf{E}_{\mu^{\setminus j}(t)}[g(U_j)] = \sum_{u_j} \mu_{u_j}(t) g(u_j)$. The parameter $\mu^i_{x_i}(t)$ appears in these terms only when i is a parent of j and u_j is consistent with x_i. In that case $\frac{\partial}{\partial \mu^i_{x_i}} \mu_{u_j} = \mu_{u_j} / \mu^i_{x_i}$. Thus,

$$\frac{\partial}{\partial \mu^i_{x_i}} \mathbf{E}_{\mu^{\setminus j}(t)}[g(U_j)] = \mathbf{E}_{\mu^{\setminus j}(t)}[g(U_j) \mid x_i] \cdot \delta_{j \in Children_i}$$

Recalling the definitions of the averaged rates

$$\overline{q}^{\,i}_{x_i,x_i|x_j}(t) = \mathbf{E}_{\mu^{\setminus i}(t)}\left[q^{i|\mathbf{Pa}_i}_{x_i,x_i|U_i} \mid x_j\right]$$

and

$$\tilde{q}^i_{x_i,y_i|x_j}(t) = \exp\left\{\mathbf{E}_{\mu^{\setminus i}(t)}\left[\ln q^{i|\mathbf{Pa}_i}_{x_i,y_i|U_i} \mid x_j\right]\right\}$$

we obtain

$$\frac{\partial}{\partial \mu^i_{x_j}} \mathbf{E}_{\mu^{\setminus j}(t)}\left[q^j_{x_j,x_j|U_j}\right] = \delta_{j \in Children_i} \overline{q}^{\,j}_{x_j,x_j|x_i}(t)$$

and

$$\frac{\partial}{\partial \mu^i_{x_j}} \mathbf{E}_{\mu^{\setminus j}(t)}\left[\ln q^j_{x_j,x_j|U_j}\right] = \delta_{j \in Children_i} \ln \tilde{q}^j_{x_j,x_j|x_i}(t).$$

Therefore the derivative of the sum over $j \neq i$ of the energy terms is

$$\psi^i_{x_i}(t) \equiv \sum_{j \in Children_i} \sum_{x_j} \left[\mu^j_{x_j}(t) \overline{q}^{\,j}_{x_j,x_j|x_i}(t) + \sum_{x_j \neq y_j} \gamma^j_{x_j,y_j}(t) \ln \tilde{q}^j_{x_j,y_j|x_i}(t)\right] \ .$$

Additionally, the derivative of the energy term for $j = i$ is $\overline{q}^{\,i}_{x_i,x_i}(t) \equiv \mathbf{E}_{\mu^{\setminus i}(t)}\left[q_{x_i,x_i|U_i}\right]$. Next, the derivative of the entropy term is $\gamma^i_{x_i,x_i}(t)/\mu^i_{x_i}(t)$. Finally, the derivative of f with respect to $\frac{d}{dt}\mu^i_{xi}(t)$ is $-\lambda^i_{x_i}(t)$. Plugging in these derivatives into Equation (23) we obtain

$$\overline{q}^{\,i}_{x_i,x_i}(t) + \psi^i_{x_i}(t) - \frac{\gamma^i_{x_i,x_i}}{\mu^i_{x_i}(t)} + \frac{d}{dt}\lambda^i_{x_i}(t) = 0 \ . \tag{24}$$

Next, the derivative w.r.t. $\gamma^i_{x_i,y_i}(t)$ gives us

$$\ln \mu^i_{x_i}(t) + \ln \tilde{q}^i_{x_i,y_i}(t) - \ln \gamma^i_{x_i,y_i}(t) + \lambda^i_{y_i}(t) - \lambda^i_{x_i}(t) = 0 \ . \tag{25}$$

Denoting $\rho^i_{x_i}(t) = \exp\{\lambda^i_{x_i}(t)\}$, Equation (25) becomes

$$\gamma^i_{x_i,y_i}(t) = \mu^i_{x_i}(t) \tilde{q}^i_{x_i,y_i}(t) \frac{\rho^i_{y_i}(t)}{\rho^i_{x_i}(t)} \ ,$$

which is the algebraic equation of γ. Using this result and the definition of $\gamma^i_{x_i,x_i}$ we have

$$\gamma^i_{x_i,x_i}(t) = -\sum_{y_i \neq x_i} \gamma^i_{x_i,y_i}(t) = -\mu^i_{x_i}(t) \sum_{x_i,y_i} \tilde{q}^i_{x_i,y_i}(t) \frac{\rho^i_{y_i}(t)}{\rho^i_{x_i}(t)}.$$

Plugging this equality into (24) and using the identity $\frac{d}{dt}\rho^i_{x_i}(t) = \frac{d}{dt}\lambda^i_{x_i}(t)\rho^i_{x_i}(t)$ gives

$$\frac{d}{dt}\rho^i_{x_i}(t) = -\rho^i_{x_i}(t)\left(\overline{q}^{\,i}_{x_i,x_i}(t) + \psi^i_{x_i}(t)\right) - \sum_{y_i \neq x_i} \tilde{q}^i_{x_i,y_i} \rho^i_{y_i}(t) \ .$$

Thus the stationary point of the Lagrangian matches the updates of (16–17). ■

Appendix G. Proof of Proposition 7

Proof We denote the time at the vertex $t_0 = (\mathsf{b}_1, T)$, the time just before as $t_1 = (\mathsf{b}_1, T-h)$ and the times just after it on each branch $t_2 = (\mathsf{b}_2, h)$ and $t_3 = (\mathsf{b}_3, h)$, as in Figure 13.

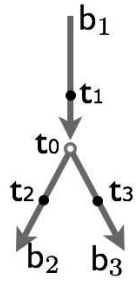


Figure 13: Branching process with discretization points of Lemma 7.

The marginals $\mu^i_{x_i}(\mathsf{b}_1, t)$ are continuous, as they are derived from the forward differential equation. To derive the propagation formula for the $\rho^i_{x_i}(t)$ requires additional care. The $\rho^i_{x_i}(t)$ have been derived from the constraints on the time-derivative of $\mu^i_{x_i}(t)$. In a vertex this constraint is threefold, as we now have the constraints on b_1

$$\frac{\mu^i_{x_i}(t_0) - \mu^i_{x_i}(t_1)}{h} = \sum_{y_i} \gamma^i_{x_i,y_i}(t_1)$$

and those on the other branches b_k for $k = 2, 3$

$$\frac{\mu^i_{x_i}(t_k) - \mu^i_{x_i}(t_0)}{h} = \sum_{y_i} \gamma^i_{x_i,y_i}(t_0) \quad .$$

The integrand of the Lagrangian corresponding to point t_0 is

$$\begin{aligned} \mathcal{L}_{|t_0} &= \tilde{\mathcal{F}}(\eta;\mathbb{Q})_{|t_0} + \lambda^0(t_1)\left(\frac{\mu^i_{x_i}(t_0) - \mu^i_{x_i}(t_1)}{h} - \sum_{y_i}\gamma^i_{x_i,y_i}(t_1)\right) \\ &- \sum_{k=2,3} \lambda^k(t_0)\left(\frac{\mu^i_{x_i}(t_k) - \mu^i_{x_i}(t_0)}{h} - \sum_{y_i}\gamma^i_{x_i,y_i}(t_0)\right) \quad , \end{aligned}$$

as this is the only integrand which involves $\mu_{x_i}(t_0)$, the derivative of the Lagrangian collapses into

$$\begin{aligned} \frac{\partial}{\partial \mu^i_{x_i}(t_0)}\mathcal{L} &= \frac{\partial}{\partial \mu^i_{x_i}(t_0)}\mathcal{L}_{|t_0} \\ &= \frac{\lambda^0(t_1)}{h} - \left(\frac{\lambda^2(t_0)}{h} + \frac{\lambda^3(t_0)}{h}\right) + \frac{\partial}{\partial \mu^i_{x_i}(t_0)}\tilde{\mathcal{F}}(\eta;\mathbb{Q})_{|t_0} = 0 \quad . \end{aligned}$$

Rearranging the previous equation and multiplying by h, we get

$$\lambda^0(t_1) = \lambda^2(t_0) + \lambda^3(t_0) + \frac{\partial}{\partial \mu^i_{x_i}(t_0)}\tilde{\mathcal{F}}(\eta;\mathbb{Q})_{|t_0} h \quad .$$

Looking at (24) we can see that as t_0 is not a leaf, and thus $\mu^i_{x_i}(t_0) > 0$ and the derivative of the functional cannot diverge. Therefore, as $h \to 0$ this term vanishes and we are left with

$$\lambda^0(t_1) = \lambda^2(t_0) + \lambda^3(t_0)$$

which after taking exponents gives (21). ■

References

C. Archambeau, M. Opper, Y. Shen, D. Cornford, and J. Shawe-Taylor. Variational inference for diffusion processes. In *Advances in Neural Information Processing Systems 20*. MIT Press, 2007.

K. L. Chung. *Markov Chains with Stationary Transition Probabilities*. Springer Verlag, Berlin, 1960.

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Comput. Intell.*, 5(3):142–150, 1989.

M. Dewar, V. Kadirkamanathan, M. Opper, and G. Sanguinetti. Parameter estimation and inference for stochastic reaction-diffusion systems: application to morphogenesis in d. melanogaster. *BMC Systems Biology*, 4(1):21, 2010.

T. El-Hay, N. Friedman, D. Koller, and R. Kupferman. Continuous time markov networks. In *Proc. Twenty-second Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006.

T. El-Hay, N. Friedman, and R. Kupferman. Gibbs sampling in factorized continuous-time markov processes. In *Proc. Twenty-fourth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

T. El-Hay, I. Cohn, N. Friedman, and R. Kupferman. Continuous-time belief propagation. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

G. Elidan, I. Mcgraw, and D. Koller. Residual belief propagation: informed scheduling for asynchronous message passing. In *Proc. Twenty-second Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006.

Y. Fan and C. R. Shelton. Sampling for approximate inference in continuous time Bayesian networks. In *Tenth International Symposium on Artificial Intelligence and Mathematics*, 2008.

Y. Fan and C. R. Shelton. Learning continuous-time social network dynamics. In *Proc. Twenty-fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

J. Felsenstein. *Inferring Phylogenies*. Sinauer, 2004.

C. W. Gardiner. *Handbook of Stochastic Methods*. Springer-Verlag, New-York, third edition, 2004.

I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Prentice-Hall, 1963.

K. Gopalratnam, H. Kautz, and D. S. Weld. Extending continuous time bayesian networks. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, pages 981–986. AAAI Press, 2005.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational approximations methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge MA, 1999.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA, 2009.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

A. Lipshtat, H. B. Perets, N. Q. Balaban, and O. Biham. Modeling of negative autoregulated genetic networks in single cells. *Gene*, 347:265, 2005.

K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

B. Ng, A. Pfeffer, and R. Dearden. Continuous time particle filtering. In *Proc. of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

U. Nodelman, C. R. Shelton, and D. Koller. Continuous time Bayesian networks. In *Proc. Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 378–387, 2002.

U. Nodelman, C. R. Shelton, and D. Koller. Learning continuous time Bayesian networks. In *Proc. Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 451–458, 2003.

U. Nodelman, C. R. Shelton, and D. Koller. Expectation maximization and complex duration distributions for continuous time Bayesian networks. In *Proc. Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 421–430, 2005a.

U. Nodelman, C. R. Shelton, and D. Koller. Expectation propagation for continuous time Bayesian networks. In *Proc. Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 431–440, 2005b.

M. Opper and G. Sanguinetti. Variational inference for Markov jump processes. In *Advances in Neural Information Processing Systems 20*. MIT Press, 2007.

M. Opper and G. Sanguinetti. Learning combinatorial transcriptional dynamics from gene expression data. *Bioinformatics*, 26(13):1623–1629, 2010.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

S. Rajaram, T. Graepel, and R. Herbrich. Poisson-networks: A model for structured point processes. In *Proc. Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, January 2005.

A. Ruttor and M. Opper. Approximate parameter inference in a stochastic reaction-diffusion model. In *Proc. Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 669–676, 2010.

G. Sanguinetti, A. Ruttor, M. Opper, and C. Archambeau. Switching regulatory models of cellular stress response. *Bioinformatics*, 25(10):1280–1286, 2009.

S. Saria, U. Nodelman, and D. Koller. Reasoning at the right time granularity. In *Proc. Twenty-third Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

A. Simma, M. Goldszmidt, J. MacCormick, P. Barham, R. Black, R. Isaacs, and R. Mortier. Ct-nor: Representing and reasoning about events in continuous time. In *Proc. Twenty-fourth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1:1–305, 2008.

J. Yu and J. L. Thorne. Dependence among sites in RNA evolution. *Mol. Biol. Evol.*, 23:1525–37, 2006.

Journal of Machine Learning Research 11 (2010) 2785-2836 Submitted 7/09; Revised 6/10; Published 10/10

Regret Bounds and Minimax Policies under Partial Monitoring

Jean-Yves Audibert* AUDIBERT@IMAGINE.ENPC.FR
Imagine, Université Paris Est
6 avenue Blaise Pascal
77455 Champs-sur-Marne, France

Sébastien Bubeck SEBASTIEN.BUBECK@INRIA.FR
SequeL Project, INRIA Lille
40 avenue Halley
59650 Villeneuve d'Ascq, France

Editor: Nicolò Cesa-Bianchi

Abstract

This work deals with four classical prediction settings, namely full information, bandit, label efficient and bandit label efficient as well as four different notions of regret: pseudo-regret, expected regret, high probability regret and tracking the best expert regret. We introduce a new forecaster, INF (Implicitly Normalized Forecaster) based on an arbitrary function ψ for which we propose a unified analysis of its pseudo-regret in the four games we consider. In particular, for $\psi(x) = \exp(\eta x) + \frac{\gamma}{K}$, INF reduces to the classical exponentially weighted average forecaster and our analysis of the pseudo-regret recovers known results while for the expected regret we slightly tighten the bounds. On the other hand with $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$, which defines a new forecaster, we are able to remove the extraneous logarithmic factor in the pseudo-regret bounds for bandits games, and thus fill in a long open gap in the characterization of the minimax rate for the pseudo-regret in the bandit game. We also provide high probability bounds depending on the cumulative reward of the optimal action.

Finally, we consider the stochastic bandit game, and prove that an appropriate modification of the upper confidence bound policy UCB1 (Auer et al., 2002a) achieves the distribution-free optimal rate while still having a distribution-dependent rate logarithmic in the number of plays.

Keywords: Bandits (adversarial and stochastic), regret bound, minimax rate, label efficient, upper confidence bound (UCB) policy, online learning, prediction with limited feedback.

1. Introduction

This section starts by defining the prediction tasks, the different regret notions that we will consider, and the different adversaries of the forecaster. We will then recap existing lower and upper regret bounds for the different settings, and give an overview of our contributions.

1.1 The Four Prediction Tasks

We consider a general prediction game where at each stage, a forecaster (or decision maker) chooses one action (or arm), and receives a reward from it. Then the forecaster receives a feedback about the rewards which he can use to make his choice at the next stage. His goal is to maximize his cumulative gain. In the simplest version, after choosing an arm the forecaster observes the rewards

*. Also at Willow, CNRS/ENS/INRIA—UMR 8548.

Parameters: the number of arms (or actions) K and the number of rounds n with $n \geq K \geq 2$.

For each round $t = 1, 2, \ldots, n$

(1) The forecaster chooses an arm $I_t \in \{1, \ldots, K\}$, possibly with the help of an external randomization.

(2) Simultaneously the adversary chooses a gain vector $g_t = (g_{1,t}, \ldots, g_{K,t}) \in [0,1]^K$ (see Section 8 for loss games or signed games).

(3) The forecaster receives the gain $g_{I_t,t}$ (without systematically observing it). He observes

- the reward vector $(g_{1,t}, \ldots, g_{K,t})$ in the **full information** game,
- the reward vector $(g_{1,t}, \ldots, g_{K,t})$ if he asks for it with the global constraint that he is not allowed to ask it more than m times for some fixed integer number $1 \leq m \leq n$. This prediction game is the **label efficient** game,
- only $g_{I_t,t}$ in the **bandit** game,
- only his obtained reward $g_{I_t,t}$ if he asks for it with the global constraint that he is not allowed to ask it more than m times for some fixed integer number $1 \leq m \leq n$. This prediction game is the **label efficient bandit** game.

Goal : The forecaster tries to maximize his cumulative gain $\sum_{t=1}^{n} g_{I_t,t}$.

Figure 1: The four prediction tasks considered in this work.

for all arms, this is the so called full information game. In the label efficient game, originally proposed by Helmbold and Panizza (1997), after choosing its action at a stage, the forecaster decides whether to ask for the rewards of the different actions at this stage, knowing that he is allowed to do it a limited number of times. Another classical setting is the bandit game where the forecaster only observes the reward of the arm he has chosen. In its original version (Robbins, 1952), this game was considered in a stochastic setting, that is, the nature draws the rewards from a fixed product-distribution. Later it was considered in an adversarial framework (Auer et al., 1995), where there is an adversary choosing the rewards on the arms. A combination of the two previous settings is the label efficient bandit game (György and Ottucsák, 2006), in which the only observed rewards are the ones obtained and asked by the forecaster, with again a limitation on the number of possible queries. These four games are described more precisely in Figure 1. Their Hannan consistency has been considered in Allenberg et al. (2006) in the case of unbounded losses. Here we will focus on regret upper bounds and minimax policies for bounded losses.

1.2 Regret and Pseudo-regret

A natural way to assess the performance of a forecaster is to compute his *regret* with respect to the best action in hindsight (see Section 7 for a more general regret in which we compare to the best

switching strategy having a fixed number of action-switches):

$$R_n = \max_{1\le i\le K} \sum_{t=1}^{n} \left(g_{i,t} - g_{I_t,t}\right).$$

A lot of attention has been drawn by the characterization of the minimax expected regret in the different games we have described. More precisely for a given game, let us write sup for the supremum over all allowed adversaries and inf for the infimum over all forecaster strategies for this game. We are interested in the quantity:

$$\inf \sup \mathbb{E} R_n,$$

where the expectation is with respect to the possible randomization of the forecaster and the adversary. Another related quantity which can be easier to handle is the *pseudo-regret*:

$$\overline{R}_n = \max_{1\le i\le K} \mathbb{E} \sum_{t=1}^{n} \left(g_{i,t} - g_{I_t,t}\right).$$

Note that, by Jensen's inequality, the pseudo-regret is always smaller than the expected regret. In Appendix D we discuss cases where the converse inequality holds (up to an additional term).

1.3 The Different Adversaries

The simplest adversary is the deterministic one. It is characterized by a fixed matrix of nK rewards corresponding to $(g_{i,t})_{1\le i\le K, 1\le t\le n}$. Another adversary is the "stochastic" one, in which the reward vectors are independent and have the same distribution.[1] This adversary is characterized by a distribution on $[0,1]^K$, corresponding to the common distribution of g_t, $t = 1,\dots,n$. A more general adversary is the fully oblivious one, in which the reward vectors are independent. Here the adversary is characterized by n distributions on $[0,1]^K$ corresponding to the distributions of $g_1,\dots,g_n$. Deterministic and stochastic adversaries are fully oblivious adversaries.

An even more general adversary is the oblivious one, in which the only constraint on the adversary is that the reward vectors are independent of the past decisions of the forecaster. The most general adversary is the one who may choose the reward vector g_t as a function of the past decisions $I_1,\dots,I_{t-1}$ (non-oblivious adversary).

1.4 Known Regret Bounds

Table 1 recaps existing lower and upper bounds on the minimax pseudo-regret and the minimax expected regret for general adversaries (i.e., possibly non-oblivious ones). For the first three lines, we refer the reader to the book (Cesa-Bianchi and Lugosi, 2006) and references within, particularly Cesa-Bianchi et al. (1997) and Cesa-Bianchi (1999) for the full information game, Cesa-Bianchi et al. (2005) for the label efficient game, Auer et al. (2002b) for the bandit game and György and Ottucsák (2006) for the label efficient bandit game. The lower bounds in the last line do not appear in the existing litterature, but we prove them in this paper. Apart from the full information game, the upper bounds are usually proved on the pseudo-regret. The upper bounds on the expected regret are obtained by using high probability bounds on the regret. The parameters of the algorithm in the

1. The term "stochastic" can be a bit misleading since the assumption is not just stochasticity but rather an i.i.d. assumption.

	$\inf\sup \overline{R}_n$		$\inf\sup \mathbb{E}R_n$	
	Lower bound	Upper bound	Lower bound	Upper bound
Full information game	$\sqrt{n\log K}$	$\sqrt{n\log K}$	$\sqrt{n\log K}$	$\sqrt{n\log K}$
Label efficient game	$n\sqrt{\frac{\log K}{m}}$	$n\sqrt{\frac{\log K}{m}}$	$n\sqrt{\frac{\log K}{m}}$	$n\sqrt{\frac{\log n}{m}}$
Bandit game	$\sqrt{nK}$	$\sqrt{nK\log K}$	$\sqrt{nK}$	$\sqrt{nK\log n}$
Bandit label efficient game	$n\sqrt{\frac{K}{m}}$	$n\sqrt{\frac{K\log K}{m}}$	$n\sqrt{\frac{K}{m}}$	$n\sqrt{\frac{K\log n}{m}}$

Table 1: Existing bounds (apart from the lower bounds in the last line which are proved in this paper) on the pseudo-regret and expected regret. Except for the full information game, there are logarithmic gaps between lower and upper bounds.

latter bounds usually depend on the confidence level δ that we want to obtain. Thus to derive bounds on the expected regret we can not integrate the deviations but rather we have to take δ of order $1/n$, which leads to the gaps involving $\log(n)$. Table 1 exhibits several logarithmic gaps between upper and lower bounds on the minimax rate, namely:

- $\sqrt{\log(K)}$ gap for the minimax pseudo-regret in the bandit game as well as the label efficient bandit game.
- $\sqrt{\log(n)}$ gap for the minimax expected regret in the bandit game as well as the label efficient bandit game.
- $\sqrt{\log(n)/\log(K)}$ gap for the minimax expected regret in the label efficient game,

1.5 Contributions of This Work

We reduce the above gaps by improving the upper bounds as shown by Table 2. Different proof techniques are used and new forecasting strategies are proposed. The most original contribution is the introduction of a new forecaster, INF (Implicitly Normalized Forecaster), for which we propose a unified analysis of its regret in the four games we consider. The analysis is original (it avoids the traditional but scope-limiting argument based on the simplification of a sum of logarithms of ratios), and allows to fill in the long open gap in the bandit problems with oblivious adversaries (and with general adversaries for the pseudo-regret notion). The analysis also applies to exponentially weighted average forecasters. It allows to prove a regret bound of order $\sqrt{nKS\log(nK/S)}$ when the forecaster's strategy is compared to a strategy allowed to switch S times between arms, while the best known bound was $\sqrt{nKS\log(nK)}$ (Auer, 2002), and achieved for a different policy.

An "orthogonal" contribution is to propose a tuning of the parameters of the forecasting policies such that the high probability regret bounds holds for any confidence level (instead of holding just for a single confidence level as in previous works). Bounds on the expected regret that are deduced from these PAC ("probably approximately correct") regret bounds are better than previous bounds by a logarithmic factor in the games with limited information (see columns on $\inf\sup \mathbb{E}R_n$ in Tables 1 and 2). The arguments to obtain these bounds are not fundamentally new and rely essentially on a careful use of deviation inequalities for supermartingales. They can be used either in the standard analysis of exponentially weighted average forecasters or in the more general context of INF.

	$\inf\sup \overline{R}_n$	$\inf\sup \mathbb{E}R_n$	High probability bound on R_n
Label efficient game		$n\sqrt{\frac{\log K}{m}}$	$n\sqrt{\frac{\log(K\delta^{-1})}{m}}$
Bandit game with fully oblivious adversary	$\sqrt{nK}$	$\sqrt{nK}$	$\sqrt{nK}\log(\delta^{-1})$
Bandit game with oblivious adversary	$\sqrt{nK}$	$\sqrt{nK}$	$\sqrt{\frac{nK}{\log K}}\log(K\delta^{-1})$
Bandit game with general adversary	$\sqrt{nK}$	$\sqrt{nK\log K}$	$\sqrt{\frac{nK}{\log K}}\log(K\delta^{-1})$
L.E. bandit with deterministic adversary	$n\sqrt{\frac{K}{m}}$	$n\sqrt{\frac{K}{m}}$	$n\sqrt{\frac{K}{m}}\log(\delta^{-1})$
L.E. bandit with oblivious adversary	$n\sqrt{\frac{K}{m}}$	$n\sqrt{\frac{K}{m}}$	$n\sqrt{\frac{K}{m\log K}}\log(K\delta^{-1})$
L.E. bandit with general adversary	$n\sqrt{\frac{K}{m}}$	$n\sqrt{\frac{K\log K}{m}}$	$n\sqrt{\frac{K}{m\log K}}\log(K\delta^{-1})$

Table 2: New regret upper bounds proposed in this work. The high probability bounds are for a policy of the forecaster that does not depend on the confidence level δ (unlike previously known high probability bounds).

Another "orthogonal" contribution is the proposal of a new biased estimate of the rewards in bandit games, which allows to achieve high probability regret bounds depending on the performance of the optimal arm: in this new bound, the factor n is replaced by $G_{\max} = \max_{i=1,\dots,n} \sum_{t=1}^{n} g_{i,t}$. If the forecaster draws I_t according to the distribution $p_t = (p_{1,t}, \dots, p_{K,t})$, then the new biased estimate of $g_{i,t}$ is $v_{i,t} = -\frac{\mathbb{1}_{I_t=i}}{\beta}\log\left(1 - \frac{\beta g_{i,t}}{p_{i,t}}\right)$. This estimate should be compared to $v_{i,t} = g_{i,t}\frac{\mathbb{1}_{I_t=i}}{p_{i,t}}$, for which bounds in terms of $G_{\max}$ exists in expectations as shown in (Auer et al., 2002b, Section 3), and to $v_{i,t} = g_{i,t}\frac{\mathbb{1}_{I_t=i}}{p_{i,t}} + \frac{\beta}{p_{i,t}}$ for some $\beta > 0$ for which high probability bounds exist but they are expressed with the n factor, and not $G_{\max}$ (see Section 6 of Auer et al., 2002b, and Section 6.8 of Cesa-Bianchi and Lugosi, 2006).

We also propose a unified proof to obtain the lower bounds in Table 1. The contribution of this proof is two-fold. First it gives the first lower bound for the label efficient bandit game. Secondly in the case of the label efficient (full information) game it is a simpler proof than the one proposed in Cesa-Bianchi et al. (2005). Indeed in the latter proof, the authors use Birgé's version of Fano's lemma to prove the lower bound for deterministic forecasters. Then the extension to non-deterministic forecasters is done by a generalization of this information lemma and a decomposition of general forecasters into a convex combination of deterministic forecasters. The benefit from this proof technique is to be able to deal with the case $K = 2$ and $K = 3$ while the basic version of Fano's lemma does not give any information in this case. Here we propose to use Pinsker's inequality for the case $K = 2$ and $K = 3$. This allows us to use the basic version of Fano's lemma and to extend the result to non-deterministic forecasters with a simple application of Fubini's Theorem.

The last contribution of this work is also independent of the previous ones and concerns the stochastic bandit game (that is the bandit game with "stochastic" adversary). We prove that a modification of UCB1, Auer et al. (2002a), attains the optimal distribution-free rate $\sqrt{nK}$ as well as the logarithmic distribution-dependent rate. The key idea, compared to previous works, is to reduce exploration of sufficiently drawn arms.

1.6 Outline

In Section 2, we describe a new class of forecasters, called INF, for prediction games. Then we present a new forecaster inside this class, called Poly INF, for which we propose a general theorem bounding its regret. A more general statement on the regret of any INF can be found in Appendix A. Exponentially weighted average forecasters are a special case of INF as shown in Section 3. In Section 4, we prove that our forecasters and analysis recover the known results for the full information game.

Section 5 contains the core contributions of the paper, namely all the regret bounds for the limited feedback games. The interest of Poly INF appears in the bandit games where it satisfies a regret bound without a logarithmic factor, unlike exponentially weighted average forecasters. Section 6 provides high probability bounds in the bandit games that depends on the cumulative reward of the optimal arm: the factor n is replaced by $\max_{1\leq i\leq K}\sum_{t=1}^{n} g_{i,t}$. In Section 7, we consider a stronger notion of regret, when we compare ourselves to a strategy allowed to switch between arms a fixed number of times. Section 8 shows how to generalize our results when one considers losses rather than gains, or signed games.

Section 9 considers a framework fundamentally different from the previous sections, namely the stochastic multi-armed bandit problem. There we propose a new forecaster, MOSS, for which we prove an optimal distribution-free rate as well as a logarithmic distribution-dependent rate.

Appendix A contains a general regret upper bound for INF and two useful technical lemmas. Appendix B contains the unified proof of the lower bounds. Appendix C contains the proofs that have not been detailed in the main body of the paper. Finally, Appendix D gathers the different results we have obtained regarding the relation between the expected regret and the pseudo-regret.

2. The Implicitly Normalized Forecaster

In this section, we define a new class of randomized policies for the general prediction game. Let us consider a continuously differentiable function $\psi : \mathbb{R}_-^* \to \mathbb{R}_+^*$ satisfying

$$\psi' > 0, \qquad \lim_{x\to-\infty} \psi(x) < 1/K, \qquad \lim_{x\to 0} \psi(x) \geq 1. \tag{1}$$

Lemma 1 *There exists a continuously differentiable function $C : \mathbb{R}_+^K \to \mathbb{R}$ satisfying for any $x = (x_1,\ldots,x_K) \in \mathbb{R}_+^K$,*

$$\max_{1\leq i\leq K} x_i < C(x) \leq \max_{1\leq i\leq K} x_i - \psi^{-1}(1/K), \tag{2}$$

and

$$\sum_{i=1}^{K} \psi(x_i - C(x)) = 1. \tag{3}$$

Proof Consider a fixed $x = (x_1,\ldots,x_K)$. The decreasing function $\phi : c \mapsto \sum_{i=1}^K \psi(x_i - c)$ satisfies

$$\lim_{c\to \max_{1\leq i\leq K} x_i} \phi(c) > 1 \qquad \text{and} \qquad \lim_{c\to+\infty} \phi(c) < 1.$$

From the intermediate value theorem, there is a unique $C(x)$ satisfying $\phi(C(x)) = 1$. From the implicit function theorem, the mapping $x \mapsto C(x)$ is continuously differentiable. ■

INF (Implicitly Normalized Forecaster):

Parameters:

- the continuously differentiable function $\psi : \mathbb{R}_-^* \to \mathbb{R}_+^*$ satisfying (1)
- the estimates $v_{i,t}$ of $g_{i,t}$ based on the (drawn arms and) observed rewards at time t (and before time t)

Let p_1 be the uniform distribution over $\{1,\dots,K\}$.

For each round $t = 1,2,\dots,$

(1) Draw an arm I_t from the probability distribution p_t.

(2) Use the observed reward(s) to build the estimate $v_t = (v_{1,t},\dots,v_{K,t})$ of $(g_{1,t},\dots,g_{K,t})$ and let: $V_t = \sum_{s=1}^{t} v_s = (V_{1,t},\dots,V_{K,t})$.

(3) Compute the normalization constant $C_t = C(V_t)$.

(4) Compute the new probability distribution $p_{t+1} = (p_{1,t+1},\dots,p_{K,t+1})$ where

$$p_{i,t+1} = \psi(V_{i,t} - C_t).$$

Figure 2: The proposed policy for the general prediction game.

The implicitly normalized forecaster (INF) is defined in Figure 2. Equality (3) makes the fourth step in Figure 2 legitimate. From (2), $C(V_t)$ is roughly equal to $\max_{1\le i\le K} V_{i,t}$. Recall that $V_{i,t}$ is an estimate of the cumulative gain at time t for arm i. This means that INF chooses the probability assigned to arm i as a function of the (estimated) regret. Note that, in spirit, it is similar to the traditional weighted average forecaster, see for example Section 2.1 of Cesa-Bianchi and Lugosi (2006), where the probabilities are proportional to a function of the difference between the (estimated) cumulative reward of arm i and the cumulative reward of the policy, which should be, for a well-performing policy, of order $C(V_t)$.

The interesting feature of the implicit normalization is the following argument, which allows to recover the results concerning the exponentially weighted average forecasters, and more interestingly to propose a policy having a regret of order $\sqrt{nK}$ in the bandit game with oblivious adversary. First note that $\sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t}v_{i,t}$ roughly evaluates the cumulative reward $\sum_{t=1}^{n} g_{I_t,t}$ of the policy. In fact, it is exactly the cumulative gain in the bandit game when $v_{i,t} = g_{i,t}\frac{\mathbb{1}_{I_t=i}}{p_{i,t}}$, and its expectation is exactly the expected cumulative reward in the full information game when $v_{i,t} = g_{i,t}$. The argument starts with an Abel transformation and consequently is "orthogonal" to the usual argument given in

the beginning of Section C.2. Letting $V_0 = 0 \in \mathbb{R}^K$. We have

$$\begin{aligned}
\sum_{t=1}^n g_{I_t,t} &\approx \sum_{t=1}^n \sum_{i=1}^K p_{i,t} v_{i,t} \\
&= \sum_{t=1}^n \sum_{i=1}^K p_{i,t}(V_{i,t} - V_{i,t-1}) \\
&= \sum_{i=1}^K p_{i,n+1} V_{i,n} + \sum_{i=1}^K \sum_{t=1}^n V_{i,t}(p_{i,t} - p_{i,t+1}) \\
&= \sum_{i=1}^K p_{i,n+1} \left(\psi^{-1}(p_{i,n+1}) + C_n\right) + \sum_{i=1}^K \sum_{t=1}^n (\psi^{-1}(p_{i,t+1}) + C_t)(p_{i,t} - p_{i,t+1}) \\
&= C_n + \sum_{i=1}^K p_{i,n+1} \psi^{-1}(p_{i,n+1}) + \sum_{i=1}^K \sum_{t=1}^n \psi^{-1}(p_{i,t+1})(p_{i,t} - p_{i,t+1}),
\end{aligned}$$

where the remarkable simplification in the last step is closely linked to our specific class of randomized algorithms. The equality is interesting since, from (2), C_n approximates the maximum estimated cumulative reward $\max_{1\le i\le K} V_{i,n}$, which should be close to the cumulative reward of the optimal arm $\max_{1\le i\le K} G_{i,n}$, where $G_{i,n} = \sum_{t=1}^n g_{i,t}$. Since the last term in the right-hand side is

$$\sum_{i=1}^K \sum_{t=1}^n \psi^{-1}(p_{i,t+1})(p_{i,t} - p_{i,t+1}) \approx \sum_{i=1}^K \sum_{t=1}^n \int_{p_{i,t}}^{p_{i,t+1}} \psi^{-1}(u)du = \sum_{i=1}^K \int_{1/K}^{p_{i,n+1}} \psi^{-1}(u)du, \tag{4}$$

we obtain

$$\max_{1\le i\le K} G_{i,n} - \sum_{t=1}^n g_{I_t,t} \lessapprox -\sum_{i=1}^K p_{i,n+1}\psi^{-1}(p_{i,n+1}) + \sum_{i=1}^K \int_{1/K}^{p_{i,n+1}} \psi^{-1}(u)du. \tag{5}$$

The right-hand side is easy to study: it depends only on the final probability vector and has simple upper bounds for adequate choices of ψ. For instance, for $\psi(x) = \exp(\eta x) + \frac{\gamma}{K}$ with $\eta > 0$ and $\gamma \in [0,1)$, which corresponds to exponentially weighted average forecasters as we will explain in Section 3, the right-hand side is smaller than $\frac{1-\gamma}{\eta}\log\left(\frac{K}{1-\gamma}\right) + \gamma C_n$. For $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ with $\eta > 0$, $q > 1$ and $\gamma \in [0,1)$, which will appear to be a fruitful choice, it is smaller than $\frac{q}{q-1}\eta K^{1/q} + \gamma C_n$. For sake of simplicity, we have been hiding the residual terms of (4) coming from the Taylor expansions of the primitive function of ψ^{-1}. However, these terms when added together (nK terms!) are not that small, and in fact constrain the choice of the parameters γ and η if one wishes to get the tightest bound.

The rigorous formulation of (5) is given in Theorem 27, which has been put in Appendix A for lack of readability. We propose here its specialization to the function $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ with $\eta > 0$, $q > 1$ and $\gamma \in [0,1)$. This function obviously satisfies conditions (1). We will refer to the associated forecasting strategy as "Poly INF". Here the (normalizing) function C has no closed form expression (this is a consequence of Abel's impossibility theorem). Actually this remark holds in general, hence the name of the general policy. However this does not lead to a major computational issue since, in the interval given by (2), $C(x)$ is the unique solution of $\phi(c) = 1$, where $\phi : c \mapsto \sum_{i=1}^K \psi(x_i - c)$ is a decreasing function. We will prove that Poly INF forecaster generates nicer probability updates than the exponentially weighted average forecasteras as, for bandits games (label efficient or not), it allows to remove the extraneous $\log K$ factor in the pseudo-regret bounds and some regret bounds.

Theorem 2 (General regret bound for Poly INF) *Let* $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ *with* $q > 1$, $\eta > 0$ *and* $\gamma \in [0,1)$. *Let* $(v_{i,t})_{1\le i\le K,\, 1\le t\le n}$ *be a sequence of nonnegative real numbers,*

$$B_t = \max_{1\le i\le K} v_{i,t}, \text{ and } B = \max_t B_t.$$

If $\gamma = 0$ *then INF satisfies:*

$$\left(\max_{1\le i\le K}\sum_{t=1}^n v_{i,t}\right) - \sum_{t=1}^n\sum_{i=1}^K p_{i,t}v_{i,t} \le \frac{q}{q-1}\eta K^{1/q} + \frac{q}{2\eta}\exp\left(2\frac{q+1}{\eta}B\right)\sum_{t=1}^n B_t^2, \tag{6}$$

and

$$\left(\max_{1\le i\le K}\sum_{t=1}^n v_{i,t}\right) - \sum_{t=1}^n\sum_{i=1}^K p_{i,t}v_{i,t} \le \frac{q}{q-1}\eta K^{1/q} + \frac{qB}{\eta}\exp\left(\frac{8qB}{\eta}\right)\sum_{t=1}^n\sum_{i=1}^K p_{i,t}v_{i,t}. \tag{7}$$

For $\gamma > 0$, *if we have* $v_{i,t} = \frac{c_t}{p_{i,t}}\mathbb{1}_{i=I_t}$ *for some random variable* c_t *taking values in* $[0,c]$ *with* $0 < c < q\eta\left(\frac{\gamma}{(q-1)K}\right)^{(q-1)/q}$, *then*

$$(1-\gamma)\left(\max_{1\le i\le K}\sum_{t=1}^n v_{i,t}\right) - (1+\gamma\zeta)\sum_{t=1}^n\sum_{i=1}^K p_{i,t}v_{i,t} \le \frac{q}{q-1}\eta K^{\frac{1}{q}}, \tag{8}$$

where

$$\zeta = \frac{1}{(q-1)K}\left(\frac{(q-1)cK\mu(1+\mu)}{2\gamma\eta}\right)^q,$$

with

$$\mu = \exp\left\{\frac{2(q+1)c}{\eta}\left(\frac{K}{\gamma}\right)^{(q-1)/q}\left(1 - \frac{c}{q\eta}\left(\frac{(q-1)K}{\gamma}\right)^{(q-1)/q}\right)^{-q}\right\}.$$

In all this work, the parameters η, q and γ will be chosen such that ζ and μ act as numerical constants. To derive concrete bounds from the above theorem, most of the work lies in relating the left-hand side with the different notions of regret we consider. This task is trivial for the pseudo-regret. To derive high probability regret bounds, deviation inequalities for supermartingales are used on top of (6) and (8) (which hold with probability one). Finally, the expected regret bounds are obtained by integration of the high probability bounds.

As long as numerical constants do not matter, one can use (7) to recover the bounds obtained from (6). The advantage of (7) over (6) is that it allows to get regret bounds where the factor n is replaced by $G_{\max} = \max_{i=1,\dots,n} G_{i,n}$.

3. Exponentially Weighted Average Forecasters

The normalization by division that weighted average forecasters perform is different from the normalization by shift of the real axis that INF performs. Nonetheless, we can recover exactly the exponentially weighted average forecasters because of the special relation of the exponential with the addition and the multiplication.

Let $\psi(x) = \exp(\eta x) + \frac{\gamma}{K}$ with $\eta > 0$ and $\gamma \in [0,1)$. Then conditions (1) are clearly satisfied and (3) is equivalent to $\exp(-\eta C(x)) = \frac{1-\gamma}{\sum_{i=1}^K \exp(\eta x_i)}$, which implies

$$p_{i,t+1} = (1-\gamma)\frac{\exp(\eta V_{i,t})}{\sum_{j=1}^K \exp(\eta V_{i,t})} + \frac{\gamma}{K}.$$

In other words, for the full information case (label efficient or not), we recover the exponentially weighted average forecaster (with $\gamma = 0$) while for the bandit game we recover EXP3. For the label efficient bandit game, it does not give us the GREEN policy proposed in Allenberg et al. (2006) but rather the straightforward modification of the exponentially weighted average forecaster to this game (György and Ottucsák, 2006). Theorem 3 below gives a unified view on this algorithm for these four games. In the following, we will refer to this algorithm as the "exponentially weighted average forecaster" whatever the game is.

Theorem 3 (Regret bound for the exponentially weighted average forecaster)
Let $\psi(x) = \exp(\eta x) + \frac{\gamma}{K}$ *with* $\eta > 0$ *and* $\gamma \in [0,1)$. *Let* $(v_{i,t})_{1\le i\le K,\, 1\le t\le n}$ *be a sequence of nonnegative real numbers,*

$$B_t = \max_{1\le i\le K} v_{i,t}, \text{ and } B = \max_{1\le t\le n} B_t.$$

Consider the increasing function $\Theta : u \mapsto \frac{e^u - 1 - u}{u^2}$ *equal to* $1/2$ *by continuity at zero. If* $\gamma = 0$ *then INF satisfies:*

$$\left(\max_{1\le i\le K} \sum_{t=1}^{n} v_{i,t}\right) - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t} \le \frac{\log K}{\eta} + \frac{\eta}{8}\sum_{t=1}^{n} B_t^2, \tag{9}$$

and

$$\left(\max_{1\le i\le K} \sum_{t=1}^{n} v_{i,t}\right) - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t} \le \frac{\log K}{\eta} + \eta B \Theta(\eta B) \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t}. \tag{10}$$

If we have

$$\gamma \ge K\eta\Theta(\eta B) \max_{i,t} p_{i,t} v_{i,t}, \tag{11}$$

then INF satisfies:

$$(1-\gamma)\left(\max_{1\le i\le K} \sum_{t=1}^{n} v_{i,t}\right) - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t} \le (1-\gamma)\frac{\log K}{\eta}. \tag{12}$$

We have the same discussion about (9) and (10) than about (6) and (7): Inequality (10) allows to prove bounds where the factor n is replaced by $G_{\max} = \max_{i=1,\dots,n} G_{i,n}$, but at the price of worsened numerical constants, when compared to (9). We illustrate this point in Theorem 4, where (13) and (14) respectively comes from (9) and (10) .

The above theorem relies on the standard argument based on the cancellation of terms in a sum of logarithms of ratios (see Section C.2). For sake of comparison, we have applied our general result for INF forecasters, that is Theorem 27 (see Appendix A). This leads to the same result with worsened constants. Precisely, $\frac{\eta}{8}$ becomes $\frac{\eta}{2}\exp(2\eta B)$ in (9) while $\Theta(\eta B)$ becomes $\frac{\exp(2B\eta)[1+\exp(2B\eta)]}{2}$ in (11). This seems to be the price for having a theorem applying to a large class of forecasters.

4. The Full Information (FI) Game

The purpose of this section is to illustrate the general regret bounds given in Theorems 2 and 3 in the simplest case, when we set $v_{i,t} = g_{i,t}$, which is possible since the rewards for all arms are observed in the full information setting. The next theorem is given explicitly to show an easy application of Inequalities (9) and (10).

Theorem 4 (Exponentially weighted average forecaster in the FI game) *Let $\psi(x) = \exp(\eta x)$ with $\eta > 0$. Let $v_{i,t} = g_{i,t}$. Then in the full information game, INF satisfies*

$$\max_{1\le i\le K} \sum_{t=1}^{n} g_{i,t} - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} g_{i,t} \le \frac{\log K}{\eta} + \frac{\eta n}{8}. \tag{13}$$

and

$$\max_{1\le i\le K} \sum_{t=1}^{n} g_{i,t} - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} g_{i,t} \le \frac{\log K}{\eta} + \eta\Theta(\eta) \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} g_{i,t}. \tag{14}$$

In particular with $\eta = \sqrt{\frac{8\log K}{n}}$, we get $\mathbb{E}R_n \le \sqrt{\frac{n}{2}\log K}$, and there exists $\eta > 0$ such that

$$\mathbb{E}R_n \le \sqrt{2\mathbb{E}G_{\max}\log K}.$$

Proof It comes from (9) and (10) since we have $B \le 1$ and $\sum_{t=1}^{n} B_t^2 \le n$. The only nontrivial result is the last inequality. It obviously holds for any η when $\mathbb{E}G_{\max} = 0$, and is achieved for $\eta = \log\left(1 + \sqrt{2(\log K)/\mathbb{E}G_{\max}}\right)$, when $\mathbb{E}G_{\max} > 0$. Indeed, by taking the expectation in (14), we get

$$\begin{aligned}\mathbb{E}\sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} g_{i,t} \ge \frac{\eta\mathbb{E}G_{\max} - \log K}{\exp(\eta) - 1} &= \log\left(1 + \sqrt{\frac{2\log K}{\mathbb{E}G_{\max}}}\right)\sqrt{\frac{(\mathbb{E}G_{\max})^3}{2\log K}} - \sqrt{\frac{\mathbb{E}G_{\max}\log K}{2}} \\ &\ge \mathbb{E}G_{\max} - 2\sqrt{\frac{\mathbb{E}G_{\max}\log K}{2}},\end{aligned}$$

where we use $\log(1+x) \ge x - \frac{x^2}{2}$ for any $x \ge 0$ in the last inequality. ■

Now we consider a new algorithm for the FI game, that is INF with $\psi(x) = \left(\frac{\eta}{-x}\right)^q$ and $v_{i,t} = g_{i,t}$.

Theorem 5 (Poly INF in the FI game) *Let $\psi(x) = \left(\frac{\eta}{-x}\right)^q$ with $\eta > 0$ and $q > 1$. Let $v_{i,t} = g_{i,t}$. Then in the full information game, INF satisfies:*

$$\max_{1\le i\le K} \sum_{t=1}^{n} g_{i,t} - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} g_{i,t} \le \frac{q}{q-1}\eta K^{1/q} + \exp\left(\frac{4q}{\eta}\right)\frac{qn}{2\eta}. \tag{15}$$

In particular with $q = 3\log K$ and $\eta = 1.8\sqrt{n\log K}$ we get

$$\mathbb{E}R_n \le 7\sqrt{n\log K}.$$

Proof It comes from (6), $q + 1 \le 2q$ and $\sum_{t=1}^{n} B_t^2 \le n$. ■

Remark 6 By using the Hoeffding-Azuma inequality (see, e.g., Lemma A.7 of Cesa-Bianchi and Lugosi, 2006), one can derive high probability bounds from (13) and (15): for instance, from (15), for any $\delta > 0$, with probability at least $1 - \delta$, Poly INF satisfies:

$$R_n \le \frac{q}{q-1}\eta K^{1/q} + \exp\left(\frac{4q}{\eta}\right)\frac{qn}{2\eta} + \sqrt{\frac{n\log(\delta^{-1})}{2}}.$$

5. The Limited Feedback Games

This section provides regret bounds for three limited feedback games: the label efficient game, the bandit game, and the mixed game, that is the label efficient bandit game.

5.1 Label Efficient Game (LE)

The variants of the LE game consider that the number of queried reward vectors is constrained either strictly or just in expectation. This section considers successively these two cases.

5.1.1 Constraint on the Expected Number of Queried Reward Vectors

As in Section 4, the purpose of this section is to show how to use INF in order to recover known minimax bounds (up to constant factors) in a slight modification of the LE game: the simple LE game, in which the requirement is that the *expected* number of queried reward vectors should be less or equal to m.

Let us consider the following policy. At each round, we draw a Bernoulli random variable Z_t, with parameter $\varepsilon = m/n$, to decide whether we ask for the gains or not. Note that we do not fulfill exactly the requirement of the LE game as we might ask a bit more than m reward vectors, but we fulfill the one of the simple LE game. We do so in order to avoid technical details and focus on the main argument of the proof. The exact LE game will be addressed in Section 5.1.2, where, in addition, we will prove bounds on the expected regret $\mathbb{E}R_n$ instead of just the pseudo-regret $\overline{R}_n$.

In this section, the estimate of $g_{i,t}$ is $v_{i,t} = \frac{g_{i,t}}{\varepsilon} Z_t$, which is observable since the rewards at time t for all arms are observed when $Z_t = 1$.

Theorem 7 (Exponentially weighted average forecaster in the simple LE game) *Let* $\psi(x) = \exp(\eta x)$ *with* $\eta = \frac{\sqrt{8m\log K}}{n}$. *Let* $v_{i,t} = \frac{g_{i,t}}{\varepsilon} Z_t$ *with* $\varepsilon = \frac{m}{n}$. *Then in the simple LE game, INF satisfies*

$$\overline{R}_n \leq n\sqrt{\frac{\log K}{2m}}.$$

Proof The first inequality comes from (9). Since we have $B_t \leq Z_t/\varepsilon$ and $v_{i,t} = \frac{g_{i,t}}{\varepsilon} Z_t$, we obtain

$$\left(\max_{1\leq i\leq K} \sum_{t=1}^{n} g_{i,t}\frac{Z_t}{\varepsilon}\right) - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} g_{i,t}\frac{Z_t}{\varepsilon} \leq \frac{\log K}{\eta} + \frac{\eta}{8\varepsilon^2}\sum_{t=1}^{n} Z_t,$$

hence, by taking the expectation of both sides,

$$\overline{R}_n = \left(\max_{1\leq i\leq K} \mathbb{E}\sum_{t=1}^{n} g_{i,t}\frac{Z_t}{\varepsilon}\right) - \mathbb{E}\sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} g_{i,t}\frac{Z_t}{\varepsilon} \leq \frac{\log K}{\eta} + \frac{n\eta}{8\varepsilon} = \frac{\log K}{\eta} + \frac{n^2\eta}{8m}.$$

Straightforward computations conclude the proof. ■

A similar result can be proved for the INF forecaster with $\psi(x) = \left(\frac{\eta}{-x}\right)^q$, $\eta > 0$ and q of order $\log K$. We do not state it since we will prove a stronger result in the next section.

5.1.2 HARD CONSTRAINT ON THE NUMBER OF QUERIED REWARD VECTORS

The goal of this section is to push the idea that by using high probability bounds as an intermediate step, one can control the expected regret $\mathbb{E}R_n = \mathbb{E}\max_{1\le i\le K}\sum_{t=1}^n \left(g_{i,t} - g_{I_t,t}\right)$ instead of just the pseudo-regret $\overline{R}_n = \max_{1\le i\le K}\mathbb{E}\sum_{t=1}^n \left(g_{i,t} - g_{I_t,t}\right)$. Most previous works have obtained results for $\overline{R}_n$. These results are interesting for oblivious opponents, that is when the adversary's choices of the rewards do not depend on the past draws and obtained rewards, since in this case Proposition 33 in Appendix D shows that one can extend bounds on the pseudo-regret $\overline{R}_n$ to the expected regret $\mathbb{E}R_n$. For non-oblivious opponents, upper bounds on $\overline{R}_n$ are rather weak statements and high probability bounds on R_n or bounds on $\mathbb{E}R_n$ are desirable. In Auer (2002) and Cesa-Bianchi and Lugosi (2006), high probability bounds on R_n have been given. Unfortunately, the policies proposed there are depending on the confidence level of the bound. As a consequence, the resulting best bound on $\mathbb{E}R_n$, obtained by choosing the policies with confidence level parameter of order $1/n$, has an extraneous $\log n$ term. Specifically, from Theorem 6.2 of Cesa-Bianchi and Lugosi (2006), one can immediately derive $\mathbb{E}R_n \le 8n\sqrt{\frac{\log(4K)+\log(n)}{m}} + 1$. The theorems of this section essentially show that the $\log n$ term can be removed.

As in Section 5.1.1, we still use a draw of a Bernoulli random variable Z_t to decide whether we ask for the gains or not. The difference is that, if $\sum_{s=1}^{t-1} Z_s \ge m$, we do not ask for the gains (as we are not allowed to do so). To avoid that this last constraint interferes in the analysis, the parameter of the Bernoulli random variable is set to $\varepsilon = \frac{3m}{4n}$ and the probability of the event $\sum_{t=1}^n Z_t > m$ is upper bounded. The estimate of $g_{i,t}$ remains $v_{i,t} = \frac{g_{i,t}}{\varepsilon} Z_t$.

Theorem 8 (Exponentially weighted average forecaster in the LE game) *Let* $\psi(x) = \exp(\eta x)$ *with* $\eta = \frac{\sqrt{m\log K}}{n}$. *Let* $v_{i,t} = \frac{g_{i,t}}{\varepsilon} Z_t$ *with* $\varepsilon = \frac{3m}{4n}$. *Then in the LE game, for any* $\delta > 0$, *with probability at least* $1-\delta$, *INF satisfies:*

$$R_n \le n\sqrt{\frac{27\log(2K\delta^{-1})}{m}},$$

and

$$\mathbb{E}R_n \le n\sqrt{\frac{27\log(6K)}{m}}.$$

Theorem 9 (Poly INF in the LE game) *Let* $\psi(x) = \left(\frac{\eta}{-x}\right)^q$ *with* $q = 3\log(2K)$ *and* $\eta = 2n\sqrt{\frac{\log(2K)}{m}}$. *Let* $v_{i,t} = \frac{g_{i,t}}{\varepsilon} Z_t$ *with* $\varepsilon = \frac{3m}{4n}$. *Then in the LE game, for any* $\delta > 0$, *with probability at least* $1-\delta$, *INF satisfies:*

$$R_n \le \left(8 - \sqrt{27}\right) n\sqrt{\frac{\log(2K)}{m}} + n\sqrt{\frac{27\log(2K\delta^{-1})}{m}},$$

and

$$\mathbb{E}R_n \le 8n\sqrt{\frac{\log(6K)}{m}}.$$

5.2 Bandit Game

This section is cut into two parts. In the first one, from Theorem 2 and Theorem 3, we derive upper bounds on the pseudo-regret $\overline{R}_n = \max_{1\le i\le K}\mathbb{E}\sum_{t=1}^n \left(g_{i,t} - g_{I_t,t}\right)$. To bound the expected regret

$\mathbb{E}R_n = \mathbb{E}\max_{1\leq i\leq K}\sum_{t=1}^n \left(g_{i,t} - g_{I_t,t}\right)$, we will then use high probability bounds on top of the use of these theorems. Since this makes the proofs more intricate, we have chosen to provide the less general results, but easier to obtain, in Section 5.2.1 and the more general ones in Section 5.2.2.

The main results here are that, by using the INF with a polynomial function ψ, we obtain an upper bound of order $\sqrt{nK}$ for $\overline{R}_n$, which imply a bound of order $\sqrt{nK}$ on $\mathbb{E}R_n$ for oblivious adversaries (Proposition 33 in Appendix D). In the general case (containing the non-oblivious opponent), we show an upper bound of order $\sqrt{nK\log K}$ on $\mathbb{E}R_n$. We conjecture that this bound cannot be improved, that is the opponent may take advantage of the past to make the player pay a regret with the extra logarithmic factor (see Remark 14).

5.2.1 BOUNDS ON THE PSEUDO-REGRET

In this section, the estimate of $g_{i,t}$ is $v_{i,t} = \frac{g_{i,t}}{p_{i,t}}\mathbb{1}_{I_t=i}$, which is observable since the reward $g_{I_t,t}$ is revealed at time t.

Theorem 10 (Exponentially weighted average forecaster in the bandit game) *Let* $\psi(x) = \exp(\eta x) + \frac{\gamma}{K}$ *with* $1 > \gamma \geq \frac{4\eta K}{5} > 0$. *Let* $v_{i,t} = \frac{g_{i,t}}{p_{i,t}}\mathbb{1}_{I_t=i}$. *Then in the bandit game, INF satisfies:*

$$\overline{R}_n \leq \frac{\log K}{\eta} + \gamma \max_{1\leq i\leq K} \mathbb{E}G_{i,n}.$$

In particular, for $\gamma = \min\left(\frac{1}{2}, \sqrt{\frac{4K\log K}{5n}}\right)$ *and* $\eta = \sqrt{\frac{5\log K}{4nK}}$, *we have*

$$\overline{R}_n \leq \sqrt{\frac{16}{5}nK\log K}.$$

Proof One simply needs to note that for $5\gamma \geq 4K\eta$, (11) is satisfied (since $B = K/\gamma$), and thus (12) can be rewritten into

$$(1-\gamma)\left(\max_{1\leq i\leq K}\sum_{t=1}^n \frac{g_{i,t}}{p_{i,t}}\mathbb{1}_{I_t=i}\right) - \sum_{t=1}^n g_{I_t,t} \leq (1-\gamma)\frac{\log K}{\eta}.$$

By taking the expectation, we get

$$\overline{R}_n \leq (1-\gamma)\frac{\log K}{\eta} + \gamma \max_{1\leq i\leq K} \mathbb{E}G_{i,n}.$$

For the numerical application, since $\overline{R}_n \leq n$, the bound is trivial $\sqrt{(4K\log K)/(5n)} < \frac{1}{2}$. Otherwise, it is a direct application of the general bound. ■

Theorem 11 (Poly INF in the bandit game) *Consider* $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ *with* $\gamma = \min\left(\frac{1}{2}, \sqrt{\frac{3K}{n}}\right)$, $\eta = \sqrt{5n}$ *and* $q = 2$. *Let* $v_{i,t} = \frac{g_{i,t}}{p_{i,t}}\mathbb{1}_{I_t=i}$. *Then in the bandit game, INF satisfies:*

$$\overline{R}_n \leq 8\sqrt{nK}.$$

Proof The bound is trivial when $\frac{1}{2} \leq \sqrt{\frac{3K}{n}}$. So we consider hereafter that $\gamma = \sqrt{\frac{3K}{n}} < \frac{1}{2}$. By taking the expectation in (8) and letting $\bar{G}_{\max} = \max_{1\leq i\leq K} \mathbb{E}G_{i,n}$, we obtain that for $\gamma > (q-1)K(q\eta)^{q/(1-q)} > 0$ (condition coming from the condition on c for (8)),

$$(1-\gamma)\bar{G}_{\max} - (1+\gamma\zeta)\mathbb{E}\sum_{t=1}^{n} g_{I_t,t} \leq \frac{q}{q-1}\eta K^{\frac{1}{q}},$$

with

$$\zeta = \frac{1}{(q-1)K}\Big(\frac{(q-1)K\mu(1+\mu)}{2\gamma\eta}\Big)^q,$$

and

$$\mu = \exp\left\{\frac{2(q+1)}{\eta}\left(\frac{K}{\gamma}\right)^{(q-1)/q}\left(1-\frac{1}{q\eta}\left(\frac{(q-1)K}{\gamma}\right)^{(q-1)/q}\right)^{-q}\right\}.$$

We thus have

$$\overline{R}_n \leq \gamma(1+\zeta)\bar{G}_{\max} + \frac{q}{q-1}\eta K^{\frac{1}{q}} \leq \gamma(1+\zeta)n + \frac{q}{q-1}\eta K^{\frac{1}{q}}.$$

The desired inequality is trivial when $\sqrt{K/n} \geq 1/8$. So we now consider that $\sqrt{K/n} < 1/8$. For $\gamma = \sqrt{3K/n}$, $\eta = \sqrt{5n}$ and $q = 2$, the condition on γ is satisfied (since $\sqrt{K/n} < 1/8$), and we have $\frac{1}{\eta}\big(\frac{K}{\gamma}\big)^{(q-1)/q} \leq 0.121$, hence $\mu \leq 2.3$, $\zeta \leq 1$ and $\overline{R}_n \leq 8\sqrt{nK}$. ■

We have arbitrarily chosen $q = 2$ to provide an explicit upper bound. More generally, it is easy to check from the proof of Theorem 11 that for any real number $q > 1$, we obtain the convergence rate $\sqrt{nK}$, provided that γ and η are respectively taken of order $\sqrt{K/n}$ and $\sqrt{nK}/K^{1/q}$.

5.2.2 HIGH PROBABILITY BOUNDS AND BOUNDS ON THE EXPECTED REGRET

Theorems 10 and 11 provide upper bounds on $\overline{R}_n = \max_{1\leq i\leq K}\mathbb{E}\sum_{t=1}^{n}\big(g_{i,t} - g_{I_t,t}\big)$. To bound $\mathbb{E}R_n = \mathbb{E}\max_{1\leq i\leq K}\sum_{t=1}^{n}\big(g_{i,t} - g_{I_t,t}\big)$, we will use high probability bounds. First we need to modify the estimates of $g_{i,t}$ by considering $v_{i,t} = \frac{g_{i,t}}{p_{i,t}}\mathbb{1}_{I_t=i} + \frac{\beta}{p_{i,t}}$ with $0 < \beta \leq 1$, as was proposed in Auer (2002),[2] or $v_{i,t} = -\frac{\mathbb{1}_{I_t=i}}{\beta}\log\big(1 - \frac{\beta g_{i,t}}{p_{i,t}}\big)$ as we propose here.

Theorem 12 (Exponentially weighted average forecaster in the bandit game) *Consider* $\psi(x) = \exp(\eta x) + \frac{\gamma}{K}$ *with* $\gamma = \min\Big(\frac{2}{3}, 2\sqrt{\frac{K\log(3K)}{n}}\Big)$ *and* $\eta = 2\sqrt{\frac{\log(3K)}{Kn}}$. *Let* $v_{i,t} = -\frac{\mathbb{1}_{I_t=i}}{\beta}\log\big(1 - \frac{\beta g_{i,t}}{p_{i,t}}\big)$ *with* $\beta = \sqrt{\frac{\log(3K)}{2Kn}}$. *Then in the bandit game, against any adversary (possibly a non-oblivious one), for any* $\delta > 0$, *with probability at least* $1-\delta$, *INF satisfies:*

$$R_n \leq 3\sqrt{nK\log(3K)} + \sqrt{\frac{2nK}{\log(3K)}}\log(K\delta^{-1}),$$

and also $\mathbb{E}R_n \leq (3+\sqrt{2})\sqrt{nK\log(3K)}$.

2. The technical reason for this modification, which may appear surprising as it introduces a bias in the estimate of $g_{i,t}$, is that it allows to have high probability upper bounds with the correct rate on the difference $\sum_{t=1}^{n} g_{i,t} - \sum_{t=1}^{n} v_{i,t}$. A second reason for this modification (but useless for this particular section) is that it allows to track the best expert (see Section 7).

This theorem is similar to Theorem 6.10 of Cesa-Bianchi and Lugosi (2006). The main difference here is that the high probability bound holds for any confidence level, and not only for a confidence level depending on the algorithm. As a consequence, our algorithm, unlike the one proposed in previous works, satisfies both a high probability bound and an expected regret bound of order $\sqrt{nK\log(K)}$.

Theorem 13 (Poly INF in the bandit game) *Let* $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ *with* $\eta = 2\sqrt{n}$, $q = 2$ *and* $\gamma = \min\left(\frac{1}{2}, 3\sqrt{\frac{K}{n}}\right)$. *Consider* $v_{i,t} = -\frac{\mathbb{I}_{I_t=i}}{\beta}\log\left(1 - \frac{\beta g_{i,t}}{p_{i,t}}\right)$ *with* $\beta = \frac{1}{\sqrt{2Kn}}$. *Then in the bandit game, against a deterministic adversary, for any* $\delta > 0$, *with probability at least* $1-\delta$, *INF satisfies:*

$$R_n \leq 9\sqrt{nK} + \sqrt{2nK}\log(\delta^{-1}). \tag{16}$$

Against an oblivious adversary, it satisfies

$$\mathbb{E}R_n \leq 10\sqrt{nK}. \tag{17}$$

Moreover in the general case (containing the non-oblivious opponent), with the following parameters $q = 2$, $\gamma = \min\left(\frac{1}{2}, 3\sqrt{\frac{K\log(3K)}{n}}\right)$, $\eta = 2\sqrt{\frac{n}{\log(3K)}}$ *and* $\beta = \sqrt{\frac{\log(3K)}{2nK}}$, *it satisfies with probability at least* $1-\delta$,

$$R_n \leq 9\sqrt{nK\log(3K)} + \sqrt{\frac{2nK}{\log(3K)}}\log(\delta^{-1}),$$

and

$$\mathbb{E}R_n \leq 9\sqrt{nK\log(3K)}.$$

Remark 14 *We conjecture that the order* $\sqrt{nK\log K}$ *of the bound on* $\mathbb{E}R_n$ *cannot be improved in the general case containing the non-oblivious opponent. Here is the main argument to support our conjecture. Consider an adversary choosing all rewards to be equal to one until time* $n/2$ *(say* n *is even to simplify). Then, let* $\hat{k}$ *denote the arm for which the estimate* $V_{i,n/2} = \sum_{1\leq t\leq n/2} v_{i,t}$ *of the cumulative reward of arm* i *is the smallest. After time* $n/2$, *all rewards are chosen to be equal to zero except for arm* $\hat{k}$ *for which the rewards are still chosen to be equal to* 1. *Since we believe that with high probability,* $\max_{1\leq i\leq K} V_{i,n/2} - \min_{j\in\{1,\ldots,K\}} V_{j,n/2} \geq \kappa\sqrt{nK\log K}$ *for some small enough* $\kappa > 0$, *it seems that the INF algorithm achieving a bound of order* $\sqrt{nK}$ *on* $\mathbb{E}R_n$ *in the oblivious setting will suffer an expected regret of order at least* $\sqrt{nK\log K}$. *While this does not prove the conjecture as one can design other algorithms, it makes the conjecture likely to hold.*

5.3 Label Efficient Bandit Game (LE Bandit)

The following theorems concern the simple LE bandit game, in which the requirement is that the *expected* number of queried rewards should be less or equal to m. We consider the following policy. At each round, we draw a Bernoulli random variable Z_t, with parameter $\varepsilon = m/n$, to decide whether the gain of the chosen arm is revealed or not. Note that this policy does not fulfil exactly the requirement of the LE bandit game as we might ask a bit more than m rewards, but, as was argued in Section 5.1.2, it can be modified in order to fulfil the hard constraint of the game. The theoretical guarantees are then the same (up to numerical constant factors).

Theorem 15 (Exponentially weighted average forecaster in the simple LE bandit game) *Let* $\psi(x) = \exp(\eta x) + \frac{\gamma}{K}$ *with* $\gamma = \min\left(\frac{1}{2}, \sqrt{\frac{4K\log K}{5m}}\right)$ *and* $\eta = \frac{1}{n}\sqrt{\frac{5m\log K}{4K}}$. *Let* $v_{i,t} = g_{i,t}\frac{\mathbb{I}_{I_t=i}}{p_{i,t}}\frac{Z_t}{\varepsilon}$ *with* $\varepsilon = \frac{m}{n}$. *Then in the simple LE bandit game, INF satisfies:*

$$\overline{R}_n \leq n\sqrt{\frac{16K\log K}{5m}}.$$

Proof One simply needs to note that for $5\gamma \geq \frac{4K\eta}{\varepsilon}$, (11) is satisfied, and thus by taking the expectation in (12), we get

$$\overline{R}_n \leq (1-\gamma)\frac{\log K}{\eta} + \gamma\mathbb{E}\max_{1\leq i\leq K} V_{i,n} \leq (1-\gamma)\frac{\log K}{\eta} + \gamma n.$$

We thus have

$$\overline{R}_n \leq \frac{n}{m}\left((1-\gamma)\frac{\log K}{\eta/\varepsilon} + \gamma m\right).$$

The numerical application for the term in parenthesis is then exactly the same as the one proposed in the proof of Theorem 10 (with n and η respectively replaced by m and η/ε). ■

Theorem 16 (Poly INF in the simple LE bandit game) *Let* $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ *with* $\gamma = \min\left(\frac{1}{2}, \sqrt{\frac{3K}{m}}\right)$, $\eta = n\sqrt{\frac{5}{m}}$ *and* $q = 2$. *Let* $v_{i,t} = g_{i,t}\frac{\mathbb{I}_{I_t=i}}{p_{i,t}}\frac{Z_t}{\varepsilon}$ *with* $\varepsilon = \frac{m}{n}$. *Then in the simple LE bandit game, INF satisfies:*

$$\overline{R}_n \leq 8n\sqrt{\frac{K}{m}}.$$

Proof By taking the expectation in (8) and letting $\bar{G}_{\max} = \max_{1\leq i\leq K}\mathbb{E}G_{i,n}$, we obtain that for $\gamma > (q-1)K(q\eta\varepsilon)^{q/(1-q)} > 0$ (condition coming from the condition on c for (8)),

$$(1-\gamma)\bar{G}_{\max} - (1+\gamma\zeta)\mathbb{E}\sum_{t=1}^{n} g_{I_t,t} \leq \frac{q}{q-1}\eta K^{\frac{1}{q}},$$

with

$$\zeta = \frac{1}{(q-1)K}\Big(\frac{(q-1)K\mu(1+\mu)}{2\gamma\eta\varepsilon}\Big)^q,$$

and

$$\mu = \exp\left\{\frac{2(q+1)}{\eta\varepsilon}\left(\frac{K}{\gamma}\right)^{(q-1)/q}\left(1 - \frac{1}{q\eta}\left(\frac{(q-1)K}{\gamma}\right)^{(q-1)/q}\right)^{-q}\right\}.$$

We thus have

$$\overline{R}_n \leq \frac{n}{m}\left(\gamma(1+\zeta)m + \frac{q}{q-1}(\eta\varepsilon)K^{\frac{1}{q}}\right).$$

The numerical application for the term in parenthesis is exactly the same than the one proposed in the proof of Theorem 11 (with n and η respectively replaced by m and $\eta\varepsilon$). ■

Both previous theorems only consider the pseudo-regret. By estimating $g_{i,t}$ differently, we obtain the following result for the regret.

Theorem 17 (Poly INF in the simple LE bandit game) *Let* $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ *with* $\eta = 2n/\sqrt{m}$, $q = 2$ *and* $\gamma = \min\left(\frac{1}{2}, 3\sqrt{\frac{K}{m}}\right)$. *Consider* $v_{i,t} = -\frac{\mathbb{I}_{I_t=i} Z_t}{\beta} \log\left(1 - \frac{\beta g_{i,t}}{\varepsilon p_{i,t}}\right)$ *with* $\beta = \frac{1}{n}\sqrt{\frac{m}{2K}}$. *Then in the simple LE bandit game, against a deterministic adversary, for any* $\delta > 0$, *with probability at least* $1-\delta$, *INF satisfies:*

$$R_n \leq 10.7n\sqrt{\frac{K}{m}} + 3.1n\sqrt{\frac{K}{m}}\log(\delta^{-1}).$$

Against an oblivious adversary, it satisfies

$$\mathbb{E}R_n \leq 13\sqrt{nK}.$$

Moreover in the general case (containing the non-oblivious opponent), with the following parameters $q = 2$, $\gamma = \min\left(\frac{1}{2}, 3\sqrt{\frac{K\log(3K)}{m}}\right)$, $\eta = 2n/\sqrt{m\log(3K)}$ *and* $\beta = \frac{1}{n}\sqrt{\frac{m\log(3K)}{2K}}$, *it satisfies with probability at least* $1-\delta$,

$$R_n \leq 10n\sqrt{\frac{K\log(3K)}{m}} + 3.5n\sqrt{\frac{K}{m\log(3K)}}\log(\delta^{-1}),$$

and

$$\mathbb{E}R_n \leq 13n\sqrt{\frac{\log(3K)}{m}}.$$

A similar result can be obtained for Exp INF, at the price of an additional logarithmic term in K against oblivious (deterministic or not) adversaries. We omit the details.

6. Regret Bounds Scaling with the Optimal Arm Rewards

In this section, we provide regret bounds for bandit games depending on the performance of the optimal arm: in these bounds, the factor n is essentially replaced by

$$G_{\max} = \max_{i=1,\ldots,n} G_{i,n},$$

where $G_{i,n} = \sum_{t=1}^n g_{i,t}$. Such a bound has been proved on the expected regret for deterministic adversaries in the seminal work of Auer et al. (2002b). Here, by using a new biased estimate of $g_{i,t}$, that is

$v_{i,t} = -\frac{\mathbb{I}_{I_t=i}}{\beta}\log\left(1 - \frac{\beta g_{i,t}}{p_{i,t}}\right)$, we obtain a bound holding with high probability and we also consider its extension to any adversary.

The bounds presented here are especially interesting when $G_{\max} \ll n$: this typically occurs in online advertizing where the different arms are the ads that can be put on the website and where the probability that a user clicks on an ad banner (and thus induces a reward to the webpage owner) is very low. For deterministic adversaries, as in the bandit game, the $\log K$ factor appearing in the exponentially weighted average forecaster regret bound disappears in the Poly INF regret bound as follows.

Theorem 18 *Let G_0 be a real number such that $G_0 \geq 81K$. Let $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ with $\eta = 2\sqrt{G_0}$, $q = 2$ and $\gamma = 3\sqrt{\frac{K}{G_0}}$. Let $v_{i,t} = -\frac{\mathbb{I}_{I_t=i}}{\beta}\log\left(1 - \frac{\beta g_{i,t}}{p_{i,t}}\right)$ with $\beta = \frac{1}{\sqrt{2KG_0}}$. Then in the bandit game, against a deterministic adversary, for any $\delta > 0$, with probability at least $1-\delta$, INF satisfies:*

$$R_n \leq 4.5\sqrt{K\frac{G_{\max}^2}{G_0}} + 4\sqrt{KG_0} + \sqrt{2KG_0}\log(\delta^{-1}). \tag{18}$$

For fully oblivious adversaries, for any $\delta > 0$, with probability at least $1-\delta$, INF satisfies:

$$R_n \leq 4.5\sqrt{K\frac{G_{\max}^2}{G_0}} + 4\sqrt{KG_0} + \sqrt{2KG_0}\log(2\delta^{-1}) + \sqrt{8\log(2K\delta^{-1})G_{\max}}. \tag{19}$$

For the choice $G_0 = n$, the high probability upper bounds are of the order of $\sqrt{nK} + \sqrt{nK}\log(\delta^{-1})$. The interest of the theorem is to provide a policy which, for small $G_{\max}$, leads to smaller regret bounds, as long as G_0 is taken much smaller than n and but not much smaller than $G_{\max}$. For deterministic adversaries, $G_{\max}$ is nonrandom, and provided that we know its order, one has interest of taking G_0 of this order. Precisely, we have the following corollary for deterministic adversaries.

Corollary 19 *Let $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ with $\eta = 2\sqrt{G_{\max}}$, $q = 2$ and $\gamma = \min\left(\frac{1}{2}, 3\sqrt{\frac{K}{G_{\max}}}\right)$. Consider $v_{i,t} = -\frac{\mathbb{I}_{I_t=i}}{\beta}\log\left(1 - \frac{\beta g_{i,t}}{p_{i,t}}\right)$ with $\beta = \frac{1}{\sqrt{2KG_{\max}}}$. Then in the bandit game, against a deterministic adversary, for any $\delta > 0$, with probability at least $1-\delta$, INF satisfies:*

$$R_n \leq 9\sqrt{KG_{\max}} + \sqrt{2KG_{\max}}\log(\delta^{-1}), \tag{20}$$

and

$$\mathbb{E}R_n \leq 10\sqrt{KG_{\max}}. \tag{21}$$

For more general adversaries than fully oblivious ones, we have the following result in which the $\log K$ factor reappears.

Theorem 20 *Let $G_0 \geq 81K\log(3K)$. Let $\psi(x) = \left(\frac{\eta}{-x}\right)^q + \frac{\gamma}{K}$ with $q = 2$, $\gamma = 3\sqrt{\frac{K\log(3K)}{G_0}}$ and $\eta = 2\sqrt{\frac{G_0}{\log(3K)}}$. Let $v_{i,t} = -\frac{\mathbb{I}_{I_t=i}}{\beta}\log\left(1 - \frac{\beta g_{i,t}}{p_{i,t}}\right)$ with $\beta = \sqrt{\frac{\log(3K)}{2KG_0}}$. Then in the bandit game, against any adversary (possibly a non-oblivious one), for any $\delta > 0$, with probability at least $1-\delta$, INF satisfies:*

$$R_n \leq \frac{9}{2}\sqrt{\frac{G_{\max}^2}{G_0}K\log(3K)} + 4\sqrt{\frac{KG_0}{\log(3K)}} + \sqrt{\frac{2KG_0}{\log(3K)}}\log(K\delta^{-1}).$$

This last result concerning Poly INF is similar to the following one concerning the exponentially weighted average forecaster: the advantage of Poly INF only appears when it allows to remove the $\log K$ factor.

Theorem 21 *Let $G_0 > 4K\log(3K)$. Let $\psi(x) = \exp(\eta x) + \frac{\gamma}{K}$ with $\gamma = 2\sqrt{\frac{K\log(3K)}{G_0}}$ and $\eta = 2\sqrt{\frac{\log(3K)}{KG_0}}$. Let $v_{i,t} = -\frac{\mathbb{I}_{I_t=i}}{\beta}\log\left(1 - \frac{\beta g_{i,t}}{p_{i,t}}\right)$ with $\beta = \sqrt{\frac{\log(3K)}{2KG_0}}$. Then in the bandit game, against any adversary (possibly a non-oblivious one), for any $\delta > 0$, with probability at least $1-\delta$, INF satisfies:*

$$R_n \leq \frac{5}{2}\sqrt{\frac{G_{\max}^2}{G_0}K\log(3K)} + \frac{1}{2}\sqrt{KG_0\log(3K)} + \sqrt{\frac{2KG_0}{\log(3K)}}\log(K\delta^{-1}).$$

7. Tracking the Best Expert in the Bandit Game

In the previous sections, the cumulative gain of the forecaster was compared to the cumulative gain of the best single expert. Here, it will be compared to more flexible strategies that are allowed to switch actions. We will use

$$v_{i,t} = g_{i,t}\frac{\mathbb{I}_{I_t=i}}{p_{i,t}} + \frac{\beta}{p_{i,t}},$$

with $0 < \beta \leq 1$. The β term introduces a bias in the estimate of $g_{i,t}$, that constrains the differences $\max_{1\leq i\leq K} V_{i,t} - \min_{1\leq j\leq K} V_{j,t}$ to be relatively small. This is the key property in order to track the best switching strategy, provided that the number of switches is not too large. A switching strategy is defined by a vector $(i_1,\ldots,i_n) \in \{1,\ldots,K\}^n$. Its size is defined by

$$\mathcal{S}(i_1,\ldots,i_n) = \sum_{t=1}^{n-1} \mathbb{I}_{i_{t+1}\neq i_t},$$

and its cumulative gain is

$$G_{(i_1,\ldots,i_n)} = \sum_{t=1}^{n} g_{i_t,t}.$$

The regret of a forecaster with respect to the best switching strategy with S switches is then given by:

$$R_n^S = \max_{(i_1,\ldots,i_n):\mathcal{S}(i_1,\ldots,i_n)\leq S} G_{(i_1,\ldots,i_n)} - \sum_{t=1}^{n} g_{I_t,t}.$$

Theorem 22 (INF for tracking the best expert in the bandit game)
Let $s = S\log\left(\frac{3nK}{S}\right) + 2\log K$ with the natural convention $S\log(3nK/S) = 0$ for $S = 0$. Let $v_{i,t} = g_{i,t}\frac{\mathbb{I}_{I_t=i}}{p_{i,t}} + \frac{\beta}{p_{i,t}}$ with $\beta = 3\sqrt{\frac{s}{nK}}$. Let $\psi(x) = \exp(\eta x) + \frac{\gamma}{K}$ with $\gamma = \min\left(\frac{1}{2}, \sqrt{\frac{Ks}{2n}}\right)$ and $\eta = \frac{1}{5}\sqrt{\frac{s}{nK}}$. Then in the bandit game, for any $0 \leq S \leq n-1$, for any $\delta > 0$, with probability at least $1-\delta$, INF satisfies:

$$R_n^S \leq 7\sqrt{nKs} + \sqrt{\frac{nK}{s}}\log(\delta^{-1}),$$

and

$$\mathbb{E}R_n^S \leq 7\sqrt{nKs}.$$

Note that for $S = 0$, we have $R_n^S = R_n$, and we recover an expected regret bound of order $\sqrt{nK\log K}$ similar to the one of Theorem 12.

Remark 23 Up to constant factors, the same bounds as the ones of Theorem 22 can be obtained (via a tedious proof not requiring new arguments than the ones presented in this work) for the INF forecaster using $\psi(x) = \frac{c_1}{K}\left(\frac{\sqrt{snK}}{-x}\right)^{c_3 s} + c_2\sqrt{\frac{s}{nK}}$, with $s = S\log\left(\frac{enK}{S}\right) + \log(2K)$ and appropriate constants c_1, c_2 and c_3.

8. Gains vs Losses, Unsigned Games vs Signed Games

To simplify, we have considered so far that the rewards were in $[0,1]$. Here is a trivial argument which shows how to transfer our analysis to loss games (i.e., games with only non-positive rewards), and more generally to signed games (i.e., games in which the rewards can be positive and negative). If the rewards, denoted now $g'_{i,t}$, are in some interval $[a,b]$ potentially containing zero, we set $g_{i,t} = \frac{g'_{i,t}-a}{b-a} \in [0,1]$. Then we can apply our analysis to:

$$\max_{i\in\{1,\ldots,K\}} \sum_{t=1}^{n} g_{i,t} - \sum_{t=1}^{n} g_{I_t,t} = \frac{1}{b-a}\left(\max_{i\in\{1,\ldots,K\}} \sum_{t=1}^{n} g'_{i,t} - \sum_{t=1}^{n} g'_{I_t,t}\right).$$

A less straightforward analysis can be done by looking at the INF algorithm directly applied to the observed rewards (and not to the renormalized rewards). In this case, as it was already noted in Remark 6.5 of Cesa-Bianchi and Lugosi (2006), the behavior of the algorithm may be very different for loss and gain games. However it can be proved that our analysis still holds up to constant factors (one has to go over the proofs and make appropriate modifications since for simplicity, we have presented the general results concerning INF under the assumptions that the estimates $v_{i,t}$ are nonnegative). In Section 6, we provide regret bounds scaling with the cumulative reward of the optimal arm. For this kind of results, renormalizing will not lead to regret bounds scaling with the cumulative reward before renormalization of the optimal arm, and consequently, the study of INF directly applied to the observed rewards is necessary. In particular, obtaining low regret bounds when the optimal arm has small cumulative loss would require appropriate modifications in the proof.

9. Stochastic Bandit Game

By considering the deterministic case when the rewards are $g_{i,t} = 1$ if $i = 1$ and $g_{i,t} = 0$ otherwise, it can be proved that the INF policies considered in Theorem 10 and Theorem 11 have a pseudo-regret lower bounded by $\sqrt{nK}$. In this simple setting, and more generally in most of the stochastic multi-armed bandit problems, one would like to suffer a much smaller regret.

We recall that in the stochastic bandit considered in this section, the rewards $g_{i,1},\ldots,g_{i,n}$ are independent and drawn from a fixed distribution ν_i on $[0,1]$ for each arm i, and the reward vectors $g_1,\ldots,g_n$ are independent.[3] The suboptimality of an arm i is then measured by $\Delta_i = \max_{1\le j\le K}\mu_j - \mu_i$ where μ_i is the mean of ν_i. We provide now a strategy achieving a $\sqrt{nK}$ regret in the worst case, and a much smaller regret as soon as the Δ_i of the suboptimal arms are much larger than $\sqrt{K/n}$.

Let $\widehat{\mu}_{i,s}$ be the empirical mean of arm i after s draws of this arm. Let $T_i(t)$ denote the number of times we have drawn arm i on the first t rounds. In this section, we propose a policy, called MOSS (Minimax Optimal Strategy in the Stochastic case), inspired by the UCB1 policy (Auer et al., 2002a). As in UCB1, each arm has an index measuring its performance, and at each round, we choose the arm having the highest index. The only difference with UCB1 is to use $\log\left(\frac{n}{Ks}\right)$ instead of $\log(t)$ at time t (see Figure 3). As a consequence, an arm that has been drawn more than n/K times has an index equal to the empirical mean of the rewards obtained from the arm, and when

3. Note that we do not assume independence of $g_{1,t},\ldots,g_{K,t}$ for each t. This assumption is usually made in the literature, but is often useless. In our work, assuming it would just have improved Proposition 36 by a constant factor, and would not have improved the constant in Theorem 24.

it has been drawn close to n/K times, the logarithmic term is much smaller than the one of UCB1, implying less exploration of this already intensively drawn arm.

MOSS (Minimax Optimal Strategy in the Stochastic case):

For an arm i, define its index $B_{i,s}$ by

$$B_{i,s} = \widehat{\mu}_{i,s} + \sqrt{\frac{\max\left(\log\left(\frac{n}{Ks}\right), 0\right)}{s}} \, .$$

for $s \geq 1$ and $B_{i,0} = +\infty$.

At time t, draw an arm maximizing $B_{i,T_i(t-1)}$.

Figure 3: The proposed policy for the stochastic bandit game.

Theorem 24 *Introduce* $\Delta = \min_{i \in \{1,\ldots,K\}:\Delta_i > 0} \Delta_i$. *MOSS satisfies*

$$\overline{R}_n \leq \frac{23K}{\Delta} \log\left(\max\left(\frac{110n\Delta^2}{K}, 10^4\right)\right), \tag{22}$$

and

$$\mathbb{E}R_n \leq 25\sqrt{nK}. \tag{23}$$

Besides, if there exists a unique arm with $\Delta_i = 0$, *we also have*

$$\mathbb{E}R_n \leq \frac{23K}{\Delta} \log\left(\max\left(\frac{140n\Delta^2}{K}, 10^4\right)\right). \tag{24}$$

The distribution-dependent bounds Inequalities (22) and (24) show the desired logarithmic dependence in n, while the distribution-free regret bound (23) has the minimax rate $\sqrt{nK}$.

Remark 25 The uniqueness of the optimal arm is really needed to have the logarithmic (in n) bound on the expected regret. This can be easily seen by considering a two-armed bandit in which both reward distributions are identical (and non degenerated). In this case, the pseudo-regret is equal to zero while the expected regret is of order $\sqrt{n}$. This reveals a fundamental difference between the expected regret and the pseudo-regret.

Remark 26 A careful tuning of the constants in front and inside the logarithmic term of $B_{i,s}$ and of the thresholds used in the proof leads to smaller numerical constants in the previous theorem, and in particular to $\sup \mathbb{E}R_n \leq 6\sqrt{nK}$. However, it makes the proof more intricate. So we will only prove (23).

Acknowledgments

Thanks to Gilles Stoltz for pointing us out Proposition 33. This work has been supported by the French National Research Agency (ANR) through the COSINUS program (ANR-08-COSI-004: EXPLO-RA project).

Appendix A. The General Regret Upper Bound of INF

Theorem 27 (INF regret upper bound) *For any nonnegative real numbers $v_{i,t}$, where $i \in \{1,\dots,K\}$ and $t \in \mathbb{N}^*$, we still use $v_t = (v_{1,t},\dots,v_{K,t})$ and $V_t = \sum_{s=1}^{t} v_t$, with the convention $V_0 = 0 \in \mathbb{R}_+^K$. Define $[V_{t-1}, V_t] = \{\lambda V_{t-1} + (1-\lambda)V_t : \lambda \in [0,1]\}$. Let*

$$B_t = \max_{1\le i\le K} v_{i,t},$$

$$\rho = \max_{1\le t\le n} \max_{v,w\in[V_{t-1},V_t],\, 1\le i\le K} \frac{\psi'(v_i - C(v))}{\psi'(w_i - C(w))},$$

and

$$A_t = \min\left(B_t^2 \sum_{i=1}^{K} \psi'\circ\psi^{-1}(p_{i,t}), (1+\rho^2)\sum_{i=1}^{K} \psi'\circ\psi^{-1}(p_{i,t}) v_{i,t}^2 \right).$$

Then the INF forecaster based on ψ satisfies:

$$\max_{1\le i\le K} V_{i,n} \le C_n \le \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t} - \sum_{i=1}^{K}\left(p_{i,n+1}\psi^{-1}(p_{i,n+1}) + \int_{p_{i,n+1}}^{1/K} \psi^{-1}(u)du \right) + \frac{\rho^2}{2}\sum_{t=1}^{n} A_t. \quad (25)$$

Proof Let us set $C_0 = C(V_0)$. The proof is divided into four steps.

First step: Rewriting $\sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t}v_{i,t}$.

We start with a simple Abel transformation:

$$\begin{aligned}
\sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t}v_{i,t} &= \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t}(V_{i,t}-V_{i,t-1}) \\
&= \sum_{i=1}^{K} p_{i,n+1}V_{i,n} + \sum_{i=1}^{K}\sum_{t=1}^{n} V_{i,t}(p_{i,t}-p_{i,t+1}) \\
&= \sum_{i=1}^{K} p_{i,n+1}\left(\psi^{-1}(p_{i,n+1})+C_n\right) + \sum_{i=1}^{K}\sum_{t=1}^{n}(\psi^{-1}(p_{i,t+1})+C_t)(p_{i,t}-p_{i,t+1}) \\
&= C_n + \sum_{i=1}^{K} p_{i,n+1}\psi^{-1}(p_{i,n+1}) + \sum_{i=1}^{K}\sum_{t=1}^{n}\psi^{-1}(p_{i,t+1})(p_{i,t}-p_{i,t+1})
\end{aligned}$$

where the last step comes from the fact that $\sum_{i=1}^{K} p_{i,t} = 1$.

Second step: A Taylor-Lagrange expansion.

For $x \in [0,1]$ we define $f(x) = \int_0^x \psi^{-1}(u)du$. Remark that $f'(x) = \psi^{-1}(x)$ and $f''(x) = 1/\psi'(\psi^{-1}(x))$. Then by the Taylor-Lagrange formula, we know that for any i, there exists $\tilde{p}_{i,t+1} \in [p_{i,t}, p_{i,t+1}]$ (with the convention $[a,b] = [b,a]$ when $a > b$) such that

$$f(p_{i,t}) = f(p_{i,t+1}) + (p_{i,t}-p_{i,t+1})f'(p_{i,t+1}) + \frac{(p_{i,t}-p_{i,t+1})^2}{2}f''(\tilde{p}_{i,t+1}),$$

or, in other words:

$$(p_{i,t}-p_{i,t+1})\psi^{-1}(p_{i,t+1}) = \int_{p_{i,t+1}}^{p_{i,t}} \psi^{-1}(u)du - \frac{(p_{i,t}-p_{i,t+1})^2}{2\psi'(\psi^{-1}(\tilde{p}_{i,t+1}))}.$$

Now by summing over t the first term on the right-hand side becomes $\int_{p_{i,n+1}}^{1/K} \psi^{-1}(u)du$. Moreover, since $x \to \psi(x - C(x))$ is continuous, there exists $W^{(i,t)} \in [V_t, V_{t+1}] \subset \mathbb{R}^K$ such that $\psi\left(W_i^{(i,t)} - C(W^{(i,t)})\right) = \tilde{p}_{i,t+1}$. Thus we have

$$\sum_{i=1}^{K}\sum_{t=1}^{n}\psi^{-1}(p_{i,t+1})(p_{i,t}-p_{i,t+1}) = \sum_{i=1}^{K}\int_{p_{i,n+1}}^{1/K}\psi^{-1}(u)du - \sum_{i=1}^{K}\sum_{t=1}^{n}\frac{(p_{i,t}-p_{i,t+1})^2}{2\psi'\left(W_i^{(i,t)} - C(W^{(i,t)})\right)}.$$

From the equality obtained in the first step, it gives

$$\begin{aligned}
C_n - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t}v_{i,t} = -\sum_{i=1}^{K}\left(p_{i,n+1}\psi^{-1}(p_{i,n+1}) + \int_{p_{i,n+1}}^{1/K}\psi^{-1}(u)du\right) \\
+ \sum_{i=1}^{K}\sum_{t=1}^{n}\frac{(p_{i,t}-p_{i,t+1})^2}{2\psi'\left(W_i^{(i,t)} - C(W^{(i,t)})\right)}.
\end{aligned}$$

Third step: The mean value theorem to compute $(p_{i,t+1}-p_{i,t})^2$.

It is now convenient to consider the functions f_i and h_i defined for any $x \in \mathbb{R}_+^K$ by

$$f_i(x) = \psi(x_i - C(x)) \quad \text{and} \quad h_i(x) = \psi'(x_i - C(x)).$$

We are going to bound $p_{i,t+1} - p_{i,t} = f_i(V_t) - f_i(V_{t-1})$ by using the mean value theorem. To do so we need to compute the gradient of f_i. First, we have

$$\frac{\partial f_i}{\partial x_j}(x) = \left(\mathbb{1}_{i=j} - \frac{\partial C}{\partial x_j}(x) \right) h_i(x).$$

Now, by definition of C, we have $\sum_{k=1}^K f_k(x) = 1$ and thus $\sum_{k=1}^K \frac{\partial f_k}{\partial x_j}(x) = 0$, which implies

$$\frac{\partial C}{\partial x_j}(x) = \frac{h_j(x)}{\sum_{k=1}^K h_k(x)} \qquad \text{and} \qquad \frac{\partial f_i}{\partial x_j}(x) = \left(\mathbb{1}_{i=j} - \frac{h_j(x)}{\sum_{k=1}^K h_k(x)} \right) h_i(x).$$

Now the mean value theorem says that there exists $V^{(i,t)} \in [V_{t-1}, V_t]$ such that

$$f_i(V_t) - f_i(V_{t-1}) = \sum_{j=1}^K v_{j,t} \frac{\partial f_i}{\partial x_j}(V^{(i,t)}).$$

Thus we have

$$\begin{aligned}(p_{i,t} - p_{i,t+1})^2 &= \left(\sum_{j=1}^K v_{j,t} \left(\mathbb{1}_{i=j} - \frac{h_j(V^{(i,t)})}{\sum_{k=1}^K h_k(V^{(i,t)})} \right) h_i(V^{(i,t)}) \right)^2 \\ &= h_i(V^{(i,t)})^2 \left(v_{i,t} - \frac{\sum_{j=1}^K v_{j,t} h_j(V^{(i,t)})}{\sum_{k=1}^K h_k(V^{(i,t)})} \right)^2.\end{aligned}$$

Fourth step: An almost variance term.

We introduce $\rho = \max_{v,w \in [V_{t-1}, V_t],\, 1 \le t \le n,\, 1 \le i \le K} \frac{h_i(v)}{h_i(w)}$. Thus we have

$$\begin{aligned}\sum_{i=1}^K \sum_{t=1}^n \frac{(p_{i,t} - p_{i,t+1})^2}{2\psi'\left(W_i^{(i,t)} - C(W^{(i,t)})\right)} &= \sum_{i=1}^K \sum_{t=1}^n \frac{h_i(V^{(i,t)})^2}{2h_i(W^{(i,t)})} \left(v_{i,t} - \frac{\sum_{j=1}^K v_{j,t} h_j(V^{(i,t)})}{\sum_{k=1}^K h_k(V^{(i,t)})} \right)^2 \\ &\le \frac{\rho^2}{2} \sum_{t=1}^n \sum_{i=1}^K h_i(V_{t-1}) \left(v_{i,t} - \frac{\sum_{j=1}^K v_{j,t} h_j(V^{(i,t)})}{\sum_{k=1}^K h_k(V^{(i,t)})} \right)^2.\end{aligned}$$

Now we need to control the term $\left(v_{i,t} - \frac{\sum_{j=1}^K v_{j,t} h_j(V^{(i,t)})}{\sum_{k=1}^K h_k(V^{(i,t)})} \right)^2$. Remark that since the function ψ is increasing we know that $h_i(x) \ge 0, \forall x$. Now since we have $0 \le v_{i,t} \le B_t$, we can simply bound this last term by B_t^2. A different bound can be obtained by using $(a-b)^2 \le a^2 + b^2$ when a and b have

the same sign:

$$\begin{aligned}\left(v_{i,t}-\frac{\sum_{j=1}^K v_{j,t}h_j(V^{(i,t)})}{\sum_{k=1}^K h_k(V^{(i,t)})}\right)^2 &\leq v_{i,t}^2+\left(\frac{\sum_{j=1}^K v_{j,t}h_j(V^{(i,t)})}{\sum_{k=1}^K h_k(V^{(i,t)})}\right)^2\\ &\leq v_{i,t}^2+\frac{\sum_{j=1}^K v_{j,t}^2 h_j(V^{(i,t)})}{\sum_{k=1}^K h_k(V^{(i,t)})}\\ &\leq v_{i,t}^2+\rho^2\frac{\sum_{j=1}^K v_{j,t}^2 h_j(V_{t-1})}{\sum_{k=1}^K h_k(V_{t-1})},\end{aligned}$$

where the first inequality comes from the fact that both terms are nonnegative and the second inequality comes from Jensen's inequality. As a consequence, we have

$$\begin{aligned}\sum_{i=1}^K h_i(V_{t-1})\left(v_{i,t}-\frac{\sum_{j=1}^K v_{j,t}h_j(V^{(i,t)})}{\sum_{k=1}^K h_k(V^{(i,t)})}\right)^2 &\leq \sum_{i=1}^K h_i(V_{t-1})v_{i,t}^2+\rho^2\sum_{j=1}^K h_j(V_{t-1})v_{j,t}^2\\ &\leq (1+\rho^2)\sum_{i=1}^K h_i(V_{t-1})v_{i,t}^2.\end{aligned}$$

We have so far proved

$$C_n \quad - \quad \sum_{t=1}^n\sum_{i=1}^K p_{i,t}v_{i,t} \quad \leq \quad -\sum_{i=1}^K\left(p_{i,n+1}\psi^{-1}(p_{i,n+1})+\int_{p_{i,n+1}}^{1/K}\psi^{-1}(u)du\right) \quad + \quad \frac{\rho^2}{2}\sum_{t=1}^n A_t.$$

The announced result is then obtained by using Inequality (2). ■

To apply successfully Theorem 27 (page 2807), we need to have tight upper bounds on ρ. The two following lemmas provide such bounds.

Lemma 28 (A simple bound on the quantity ρ of Theorem 27) *Let ψ be a convex function satisfying* (1) *and assume that there exists $B>0$ such that $\forall i,j,t\ |v_{i,t}-v_{j,t}|\leq B$. Then:*

$$\rho=\max_{1\leq t\leq n}\ \max_{v,w\in[V_{t-1},V_t],\,1\leq i\leq K}\frac{\psi'(v_i-C(v))}{\psi'(w_i-C(w))}\leq\sup_{x\in(-\infty,\psi^{-1}(1)]}\exp\left(B\frac{\psi''}{\psi'}(x)\right).$$

Proof Let $h_i(x)=\psi'(x_i-C(x)), m_i(x)=\psi''(x_i-C(x))$. For $\alpha\in[0,1]$ we note

$$\varphi(\alpha)=\log\left\{h_i(V_{t-1}+\alpha(V_t-V_{t-1}))\right\}.$$

Remark that we should rather note this function $\varphi_{i,t}(\alpha)$ but for sake of simplicity we omit this dependency. With these notations we have $\rho=\max_{\alpha,\beta\in[0,1];\,1\leq t\leq n,\,1\leq i\leq K}\exp(\varphi(\alpha)-\varphi(\beta))$. By the mean value theorem for any $\alpha,\beta\in[0,1]$ there exists $\xi\in[0,1]$ such that $\varphi(\alpha)-\varphi(\beta)=(\alpha-\beta)\varphi'(\xi)$. Now with the calculus done in the third step of the proof of Theorem 27 and using the notations $h_i:=h_i(V_{t-1}+\xi(V_t-V_{t-1})), m_i:=m_i(V_{t-1}+\xi(V_t-V_{t-1}))$ we obtain

$$\varphi'(\xi)=\sum_{j=1}^K(V_{j,t}-V_{j,t-1})\left(\mathbb{1}_{i=j}-\frac{h_j}{\sum_{k=1}^K h_k}\right)\frac{m_i}{h_i}=\sum_{j=1}^K\frac{(v_{i,t}-v_{j,t})h_j}{\sum_{k=1}^K h_k}\frac{m_i}{h_i}.$$

Thus we get

$$|\varphi'(\xi)| \leq \max_{1\leq i,j\leq K} |v_{i,t} - v_{j,t}| \sup_{v\in[V_{t-1},V_t]} \frac{\psi''}{\psi'}(v_i - C(v)).$$

Moreover, using that $x \to \psi(x - C(x))$ is continuous we know that there exists $\tilde{p}_{i,t+1} \in [p_{i,t}, p_{i,t+1}]$ such that $\tilde{p}_{i,t+1} = \psi(v_i - C(v))$ and thus $v_i - C(v) = \psi^{-1}(\tilde{p}_{i,t+1})$. This concludes the proof. ■

Lemma 29 (An other bound on the quantity ρ of Theorem 27) *Let ψ be a function satisfying* (1) *and assume that there exists $c > 0$ such that $0 \leq v_{i,t} \leq \frac{c}{p_{i,t}} \mathbb{I}_{i=I_t}$. We also assume that ψ'/ψ is a nondecreasing function and that there exists $a > 1$ such that $\psi\left(x + \frac{c}{\psi(x)}\right) \leq a\psi(x)$. Then:*

$$\rho \leq \sup_{x\in(-\infty,\psi^{-1}(1)]} \exp\left(ac\frac{\psi''}{\psi\times\psi'}(x)\right).$$

Proof We extract from the previous proof that $\rho \leq \max_{\xi\in[0,1];\, 1\leq t\leq n,\, 1\leq i\leq K} \exp(|\varphi'(\xi)|)$ where

$$\varphi'(\xi) = \sum_{j=1}^{K} \frac{(v_{i,t} - v_{j,t})h_j}{\sum_{k=1}^{K} h_k} \frac{m_i}{h_i}.$$

Note that, since the functions ψ and ψ'/ψ are nondecreasing, the function ψ is convex, hence $\psi'' \geq 0$ and $m_i \geq 0$. Now using our assumption on $v_{i,t}$ and since $p_{i,t} = f_i(V_{t-1})$, if $i \neq I_t$ we have:

$$|\varphi'(\xi)| \leq \frac{c\frac{h_{I_t}}{f_{I_t}(V_{t-1})}}{\sum_{k=1}^{K} h_k} \frac{m_i}{h_i} \leq c\frac{f_{I_t}(V_{t-1} + \xi(V_t - V_{t-1}))}{f_{I_t}(V_{t-1})} \times \frac{h_{I_t}}{f_{I_t}(V_{t-1} + \xi(V_t - V_{t-1}))} \times \frac{m_i}{h_i} \times \frac{1}{h_{I_t} + h_i}.$$

Noticing that for any x, y in $\mathbb{R}^*_-$, $\frac{\frac{\psi'}{\psi}(x)\times\frac{\psi''}{\psi'}(y)}{\psi'(x)+\psi'(y)} \leq \frac{\psi''(y)}{\psi'(y)\psi(y)}$, we obtain

$$|\varphi'(\xi)| \leq c\frac{f_{I_t}(V_{t-1} + \xi(V_t - V_{t-1}))}{f_{I_t}(V_{t-1})} \frac{m_i}{h_i \times f_i(V_{t-1} + \xi(V_t - V_{t-1}))}.$$

On the other hand if $i = I_t$ then

$$|\varphi'(\xi)| \leq \frac{c}{f_i(V_{t-1})} \frac{m_i}{h_i}.$$

To finish we only have to prove that $f_{I_t}(V_{t-1} + \xi(V_t - V_{t-1})) \leq a f_{I_t}(V_{t-1})$. Since ψ is increasing it is enough to prove that $f_{I_t}(V_t) \leq a f_{I_t}(V_{t-1})$ which is equivalent to

$$\psi(V_{I_t,t-1} + v_{I_t,t} - C_t) \leq a\psi(V_{I_t,t-1} - C_{t-1}).$$

Since $0 \leq v_{i,t} \leq \frac{c}{p_{i,t}} \mathbb{I}_{i=I_t}$ and C is an increasing function in each of its argument it is enough to prove

$$\psi\left(V_{I_t,t-1} - C_{t-1} + \frac{c}{\psi(V_{I_t,t-1} - C_{t-1})}\right) \leq a\psi(V_{I_t,t-1} - C_{t-1})$$

which is true by hypothesis on ψ. ■

Appendix B. Lower Bounds

In this section we propose a simple unified proof to derive lower bounds on the pseudo-regret in the four problems that we consider.

Theorem 30 *Let $m \geq K$. Let* sup *represents the supremum taken over all oblivious adversaries and* inf *the infimum taken over all forecasters, then the following holds true in the label efficient game:*[4]

$$\inf \sup \overline{R}_n \geq 0.03 n \sqrt{\frac{\log(K)}{m}}.$$

and in the label efficient bandit game we have:

$$\inf \sup \overline{R}_n \geq 0.04 n \sqrt{\frac{K}{m}}.$$

Proof *First step: Definitions.*

We consider a set of K oblivious adversaries. The i^{th} adversary selects its gain vectors as follows: For any $t \in \{1, \ldots, n\}, g_{i,t} \sim Ber\left(\frac{1+\varepsilon}{2}\right)$ and for $j \neq i, g_{j,t} \sim Ber\left(\frac{1-\varepsilon}{2}\right)$. We note $\mathbb{E}_i$ when we integrate with respect to the reward generation process of the i^{th} adversary. We focus on the label efficient versions of the full information and bandits games since by taking $m = n$ we recover the traditional games.

Until the fifth step we consider a deterministic forecaster, that is he does not have access to an external randomization. Let $q_n = (q_{1,n}, \ldots, q_{K,n})$ be the empirical distribution of plays over the arms defined by:

$$q_{i,n} = \frac{\sum_{t=1}^n \mathbb{1}_{I_t = i}}{n}.$$

Let J_n be drawn according to q_n. We note $\mathbb{P}_i$ the law of J_n when the forecaster plays against the i^{th} adversary. Remark that we have $\mathbb{P}_i(J_n = j) = \mathbb{E}_i \frac{1}{n} \sum_{t=1}^n \mathbb{1}_{I_t = j}$, hence, against the i^{th} adversary we have:

$$\overline{R}_n = \mathbb{E}_i \sum_{t=1}^n (g_{i,t} - g_{I_t,t}) = \varepsilon n \sum_{j \neq i} \mathbb{P}_i(J_n = j) = \varepsilon n (1 - \mathbb{P}_i(J_n = i)),$$

which implies (since a maximum is larger than a mean)

$$\sup \overline{R}_n \geq \varepsilon n \left(1 - \frac{1}{K} \sum_{i=1}^K \mathbb{P}_i(J_n = i)\right). \tag{26}$$

Second step: Information inequality.

Let $\mathbb{P}_0$ (respectively $\mathbb{P}_{K+1}$) be the law of J_n against the adversary drawing all its losses from the Bernoulli of parameter $\frac{1-\varepsilon}{2}$ (respectively $\frac{1-\varepsilon}{2} + \frac{\varepsilon}{K}$), we call it the 0^{th} adversary (respectively the $(K+1)^{th}$ adversary). Now we use either Pinsker's inequality which gives:

$$\mathbb{P}_i(J_n = i) \leq \mathbb{P}_0(J_n = i) + \sqrt{\frac{1}{2} \mathrm{KL}(\mathbb{P}_0, \mathbb{P}_i)},$$

4. Slightly better numerical constants can be obtained with a more careful optimization in step four of the proof.

and thus (thanks to the concavity of the square root)

$$\frac{1}{K}\sum_{i=1}^{K}\mathbb{P}_i(J_n=i)\leq\frac{1}{K}+\sqrt{\frac{1}{2K}\sum_{i=1}^{K}\mathrm{KL}(\mathbb{P}_0,\mathbb{P}_i)}; \tag{27}$$

or Fano's lemma:

$$\frac{1}{K}\sum_{i=1}^{K}\mathbb{P}_i(J_n=i)\leq\frac{\log 2+\frac{1}{K}\sum_{i=1}^{K}\mathrm{KL}(\mathbb{P}_i,\mathbb{P}_{K+1})}{\log(K-1)}. \tag{28}$$

We will use (28) for the full information games when $K>3$ and (27) the bandits games and the full information games with $K\in\{2,3\}$.

Third step: Computation of $\mathrm{KL}(\mathbb{P}_0,\mathbb{P}_i)$ *and* $\mathrm{KL}(\mathbb{P}_i,\mathbb{P}_{K+1})$ *with the chain rule for Kullback-Leibler divergence*.

Remark that since the forecaster is deterministic, the sequence of observed rewards (up to time n) W_n ($W_n\in\{0,1\}^{mK}$ for the full information label efficient game and $W_n\in\{0,1\}^m$ for the label efficient bandit game) uniquely determines the empirical distribution of plays q_n, and in particular the law of J_n conditionally to W_n is the same for any adversary. Thus, if for $i\in\{0,\dots,K+1\}$ we note $\mathbb{P}_i^n$ the law of W_n when the forecaster plays against the i^{th} adversary, then one can easily prove that

$$\mathrm{KL}(\mathbb{P}_0,\mathbb{P}_i)\leq\mathrm{KL}(\mathbb{P}_0^n,\mathbb{P}_i^n),\ \text{ and }\ \mathrm{KL}(\mathbb{P}_i,\mathbb{P}_{K+1})\leq\mathrm{KL}(\mathbb{P}_i^n,\mathbb{P}_{K+1}^n).$$

Now we use the Chain rule for Kullback-Leibler divergence iteratively to introduce the laws $\mathbb{P}_i^t$ of the observed rewards W_t up to time t. We also note $Z_t=1$ if some rewards are revealed at the end of round t and $Z_t=0$ otherwise. With these notations we have in the full information games, for $K>3$,

$$\begin{aligned}
&\mathrm{KL}(\mathbb{P}_i^n,\mathbb{P}_{K+1}^n)\\
&\quad=\ \mathrm{KL}(\mathbb{P}_i^1,\mathbb{P}_{K+1}^1)+\sum_{t=2}^{n}\sum_{w_{t-1}}\mathbb{P}_i^{t-1}(w_{t-1})\mathrm{KL}(\mathbb{P}_i^t(.|w_{t-1}),\mathbb{P}_{K+1}^t(.|w_{t-1}))\\
&\quad=\ \mathrm{KL}(\mathbb{P}_i^1,\mathbb{P}_{K+1}^1)\\
&\qquad+\sum_{t=2}^{n}\left\{\sum_{w_{t-1}:Z_t=1}\mathbb{P}_i^{t-1}(w_{t-1})\left[\mathrm{KL}\left(\frac{1+\varepsilon}{2},\frac{1-\varepsilon}{2}+\frac{\varepsilon}{K}\right)+(K-1)\mathrm{KL}\left(\frac{1-\varepsilon}{2},\frac{1-\varepsilon}{2}+\frac{\varepsilon}{K}\right)\right]\right\}\\
&\quad=\ \left[\mathrm{KL}\left(\frac{1+\varepsilon}{2},\frac{1-\varepsilon}{2}+\frac{\varepsilon}{K}\right)+(K-1)\mathrm{KL}\left(\frac{1-\varepsilon}{2},\frac{1-\varepsilon}{2}+\frac{\varepsilon}{K}\right)\right]\mathbb{E}_i\sum_{t=1}^{n}Z_t\\
&\quad\leq\ m\left[\mathrm{KL}\left(\frac{1+\varepsilon}{2},\frac{1-\varepsilon}{2}+\frac{\varepsilon}{K}\right)+(K-1)\mathrm{KL}\left(\frac{1-\varepsilon}{2},\frac{1-\varepsilon}{2}+\frac{\varepsilon}{K}\right)\right].
\end{aligned}$$

Summing and plugging this into (28) we obtain for the full information games:

$$\frac{1}{K}\sum_{i=1}^{K}\mathbb{P}_i(J_n=i)\leq\frac{\log 2+m\mathrm{KL}\left(\frac{1+\varepsilon}{2},\frac{1-\varepsilon}{2}+\frac{\varepsilon}{K}\right)+m(K-1)\mathrm{KL}\left(\frac{1-\varepsilon}{2},\frac{1-\varepsilon}{2}+\frac{\varepsilon}{K}\right)}{\log(K-1)}. \tag{29}$$

In the bandits games we have:

$$\begin{aligned}
&\mathrm{KL}(\mathbb{P}_0^n, \mathbb{P}_i^n) \\
&= \mathrm{KL}(\mathbb{P}_0^1, \mathbb{P}_i^1) + \sum_{t=2}^{n} \sum_{w_{t-1}} \mathbb{P}_0^{t-1}(w_{t-1}) \mathrm{KL}(\mathbb{P}_0^t(.|w_{t-1}), \mathbb{P}_i^t(.|w_{t-1})) \\
&= \mathrm{KL}(\mathbb{P}_0^1, \mathbb{P}_i^1) + \sum_{t=2}^{n} \sum_{w_{t-1}: Z_t=1, I_t=i} \mathbb{P}_0^{t-1}(w_{t-1}) \mathrm{KL}\left(\frac{1-\varepsilon}{2}, \frac{1+\varepsilon}{2}\right) \\
&= \mathrm{KL}\left(\frac{1-\varepsilon}{2}, \frac{1+\varepsilon}{2}\right) \mathbb{E}_0 \sum_{t=1}^{n} \mathbb{1}_{Z_t=1, I_t=i}.
\end{aligned}$$

Summing and plugging this into (27) we obtain for the bandits games:

$$\frac{1}{K} \sum_{i=1}^{K} \mathbb{P}_i(J_n = i) \leq \frac{1}{K} + \sqrt{\frac{m}{2K} \mathrm{KL}\left(\frac{1-\varepsilon}{2}, \frac{1+\varepsilon}{2}\right)}. \tag{30}$$

Note that with the same reasoning we obtain for the full information games:

$$\frac{1}{K} \sum_{i=1}^{K} \mathbb{P}_i(J_n = i) \leq \frac{1}{K} + \sqrt{\frac{m}{2} \mathrm{KL}\left(\frac{1-\varepsilon}{2}, \frac{1+\varepsilon}{2}\right)}. \tag{31}$$

Fourth step: Conclusion for deterministic forecasters.

To conclude the proof for deterministic forecaster one needs to plug in (29) (for the full information games with $K > 3$) or (31) (for the full information games with $K \in \{2, 3\}$) or (30) (for the bandits games) in (26) along with straightforward computations and the following simple formula:

$$\mathrm{KL}(p, q) \leq \frac{(p-q)^2}{q(1-q)}.$$

Fifth step: Fubini's theorem to handle non-deterministic forecasters.

Now let us consider a randomized forecaster. Denote by $\mathbb{E}_{reward,i}$ the expectation with respect to the reward generation process of the i^{th} adversary, $\mathbb{E}_{rand}$ the expectation with respect to the randomization of the forecaster and $\mathbb{E}_i$ the expectation with respect to both processes. Then one has (thanks to Fubini's Theorem),

$$\frac{1}{K} \sum_{i=1}^{K} \mathbb{E}_i \sum_{t=1}^{n} (g_{i,t} - g_{I_t,t}) = \mathbb{E}_{rand} \frac{1}{K} \sum_{i=1}^{K} \mathbb{E}_{reward,i} \sum_{t=1}^{n} (g_{i,t} - g_{I_t,t}).$$

Now remark that if we fix the realization of the forecaster's randomization then the results of the previous steps apply and in particular we can lower bound $\frac{1}{K} \sum_{i=1}^{K} \mathbb{E}_{reward,i} \sum_{t=1}^{n} (g_{i,t} - g_{I_t,t})$ as before. ■

Appendix C. Proofs

This section gathers the proofs that have not been provided so far.

C.1 Proof of Theorem 2 (page 2793)

The proof relies on combining Theorem 27 (page 2807) with Lemma 28 (page 2810) for $\gamma = 0$, and with Lemma 29 (page 2811) for $\gamma > 0$.

We make use of Theorem 27 and start with straightforward computations to bound the first sum in (25). We have $\psi^{-1}(x) = -\eta(x-\gamma/K)^{-1/q}$ which admits as a primitive $\int \psi^{-1}(u)du = \frac{-\eta}{1-1/q}(u-\gamma/K)^{1-1/q}$. Thus one immediately gets

$$\int_{p_{i,n+1}}^{1/K} (-\psi^{-1})(u)du \leq \frac{\eta}{1-1/q}\frac{1}{K^{1-1/q}} - \eta(p_{i,n+1}-\gamma/K)^{1-1/q}$$

and

$$p_{i,n+1}(-\psi^{-1})(p_{i,n+1}) = -\frac{\gamma}{K}\psi^{-1}(p_{i,n+1}) + \eta(p_{i,n+1}-\gamma/K)^{1-1/q}.$$

Summing over i proves that

$$-\sum_{i=1}^{K}\left(p_{i,n+1}\psi^{-1}(p_{i,n+1}) + \int_{p_{i,n+1}}^{1/K}\psi^{-1}(u)du\right) \leq \frac{q}{q-1}\eta K^{1/q} - \frac{\gamma}{K}\sum_{i=1}^{K}\psi^{-1}(p_{i,n+1}).$$

With the notations of Theorem 27, we need now to bound ρ and A_t. First we deal with the case $\gamma = 0$. Lemma 28 (page 2810) implies $\rho \leq \exp(B(q+1)/\eta)$ since we have $\frac{\psi''}{\psi'}(x) = \frac{q+1}{-x} = \frac{q+1}{\eta}\psi(x)^{1/q}$. The proof of (6) is concluded by $\psi' = \frac{q}{\eta}\psi^{(q+1)/q}$, and

$$A_t \leq B_t^2\sum_{i=1}^{K}\psi'\circ\psi^{-1}(p_{i,t}) = B_t^2\sum_{i=1}^{K}\frac{q}{\eta}p_{i,t}^{(q+1)/q} \leq \frac{q}{\eta}B_t^2.$$

For (7), the term A_t is controlled differently:

$$A_t \leq (1+\rho^2)\sum_{i=1}^{K}\psi'\circ\psi^{-1}(p_{i,t})v_{i,t}^2 = \frac{q}{\eta}(1+\rho^2)\sum_{i=1}^{K}p_{i,t}^{(q+1)/q}v_{i,t}^2 \leq \frac{qB}{\eta}(1+\rho^2)\sum_{i=1}^{K}p_{i,t}v_{i,t}.$$

Now we have already seen that $\rho \leq \exp(B(q+1)/\eta)$, hence $\rho^2(1+\rho^2) \leq 2\exp(8Bq/\eta)$, which leads to (7).

The case $\gamma > 0$ is more intricate. This is why we restrict ourselves to a specific form for the estimates $v_{i,t}$, see the assumption in Theorem 2. We start by using Lemma 29 (page 2811) to prove that $\rho \leq \mu$. First we have $\frac{\psi''}{\psi'} = \frac{q+1}{\eta}(\psi-\gamma/K)^{1/q} \leq \frac{q+1}{\eta}\psi^{1/q}$. Besides, for any $a \geq b \geq d$ we have $\frac{a}{b} \leq \frac{a-d}{b-d}$ and thus for any $x < 0$, we have

$$\frac{\psi\left(x+\frac{c}{\psi(x)}\right)}{\psi(x)} \leq \frac{\psi\left(x+\frac{c}{\psi(x)}\right)-\frac{\gamma}{K}}{\psi(x)-\frac{\gamma}{K}} = \left(1-\frac{c}{-x\psi(x)}\right)^{-q} \leq \left(1-\frac{c}{q\eta}\left(\frac{(q-1)K}{\gamma}\right)^{(q-1)/q}\right)^{-q}.$$

Thus Lemma 29 gives us

$$\rho^2 \leq \exp\left\{\frac{2(q+1)c}{\eta}\left(\frac{K}{\gamma}\right)^{(q-1)/q}\left(1-\frac{c}{q\eta}\left(\frac{(q-1)K}{\gamma}\right)^{(q-1)/q}\right)^{-q}\right\} = \mu.$$

Next we use $\psi' = \frac{q}{\eta}(\psi - \gamma/K)^{(q+1)/q}$ and the form of $v_{i,t}$ to get

$$A_t \leq (1+\rho^2)\sum_{i=1}^{K} \psi' \circ \psi^{-1}(p_{i,t}) v_{i,t}^2 \leq \frac{q(1+\mu)}{\eta} \sum_{i=1}^{K} p_{i,t}^{(q+1)/q} v_{i,t}^2 = \frac{q(1+\mu)}{\eta} p_{I_t,t}^{(1-q)/q} c_t^2.$$

Let $\zeta' = \frac{qc\mu(1+\mu)}{2\eta}$. From Theorem 27, we get

$$\begin{aligned}
C_n - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t} &\leq \frac{q}{q-1}\eta K^{1/q} + \gamma C_n + \frac{\rho^2}{2}\sum_{t=1}^{n} A_t - \frac{\gamma}{K}\sum_{t=1}^{n}\sum_{i=1}^{K} v_{i,t} \\
&\leq \frac{q}{q-1}\eta K^{1/q} + \gamma C_n + \sum_{t=1}^{n} c_t \left(\zeta' p_{I_t,t}^{(1-q)/q} - \frac{\gamma}{K p_{I_t,t}} \right) \\
&\leq \frac{q}{q-1}\eta K^{1/q} + \gamma C_n + \max_{u>0} \left(\zeta' u^{(1-q)/q} - \frac{\gamma}{Ku} \right) \sum_{t=1}^{n} c_t \\
&= \frac{q}{q-1}\eta K^{1/q} + \gamma C_n + \frac{\gamma}{(q-1)K} \left(\frac{(q-1)\zeta' K}{q\gamma} \right)^q \sum_{t=1}^{n} c_t \\
&= \frac{q}{q-1}\eta K^{1/q} + \gamma C_n + \gamma\zeta \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t}.
\end{aligned}$$

The proof of (8) is concluded by using Inequality (2).

C.2 Proof of Theorem 3 (page 2794)

We have

$$\begin{aligned}
\sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t} &= \sum_{t=1}^{n} \mathbb{E}_{k\sim p_t} v_{k,t} \\
&= \frac{1-\gamma}{\eta} \sum_{t=1}^{n} \left(\log \mathbb{E}_{i\sim q_t} \exp(\eta v_{i,t}) - \log \left[\exp\left(-\frac{\eta}{1-\gamma} \mathbb{E}_{k\sim p_t} v_{k,t} \right) \mathbb{E}_{i\sim q_t} \exp(\eta v_{i,t}) \right] \right) \\
&= \frac{1-\gamma}{\eta} \left(S - \sum_{t=1}^{n} \log(D_t) \right),
\end{aligned}$$

where

$$\begin{aligned}
S &= \sum_{t=1}^{n} \log \mathbb{E}_{i\sim q_t} \exp(\eta v_{i,t}) \\
&= \sum_{t=1}^{n} \log \left(\frac{\sum_{i=1}^{K} \exp(\eta V_{i,t})}{\sum_{i=1}^{K} \exp(\eta V_{i,t-1})} \right) = \log \left(\frac{\sum_{i=1}^{K} \exp(\eta V_{i,n})}{K} \right) \geq \eta \max_{1\leq i\leq K} V_{i,n} - \log K
\end{aligned}$$

and

$$D_t = \exp\left(-\frac{\eta}{1-\gamma} \mathbb{E}_{k\sim p_t} v_{k,t} \right) \mathbb{E}_{i\sim q_t} \exp(\eta v_{i,t})$$

When $\gamma = 0$, since $0 \leq v_{i,t} \leq B_t$, by applying Hoeffding's inequality, we get $\log(D_t) \leq \frac{\eta^2 B_t^2}{8}$, hence Inequality (9). For $\gamma = 0$, we can also use Lemma 35 and obtain $\log(D_t) \leq \eta^2 B \Theta(\eta B) \mathbb{E}_{i\sim p_t} v_{i,t}$,

hence Inequality (10). For γ satisfying (11), we have

$$\begin{aligned} D_t &\leq \exp\left(-\frac{\eta}{1-\gamma}\mathbb{E}_{k\sim p_t} v_{k,t}\right)\mathbb{E}_{i\sim q_t}\left(1+\eta v_{i,t}+\Theta(\eta B)\eta^2 v_{i,t}^2\right) \qquad (32)\\ &= \exp\left(-\frac{\eta}{1-\gamma}\mathbb{E}_{k\sim p_t} v_{k,t}\right)\left(1+\frac{\eta}{1-\gamma}\mathbb{E}_{i\sim p_t} v_{i,t}-\eta\frac{\gamma\sum_{i=1}^K v_{i,t}}{K(1-\gamma)}+\Theta(\eta B)\eta^2\mathbb{E}_{i\sim q_t} v_{i,t}^2\right)\\ &\leq \exp\left(-\frac{\eta}{1-\gamma}\mathbb{E}_{k\sim p_t} v_{k,t}\right)\left(1+\frac{\eta}{1-\gamma}\mathbb{E}_{i\sim p_t} v_{i,t}\right) \qquad (33)\\ &\leq 1. \end{aligned}$$

To get (32), we used that Θ is an increasing function and that $\eta v_{i,t}\leq \eta B$. To get (33), we noticed that it is trivial when $\max_{i,t} p_{i,t}v_{i,t}=0$, and that otherwise, we have

$$\frac{\gamma\sum_{i=1}^K v_{i,t}}{K(1-\gamma)}\geq\frac{\gamma\sum_{i=1}^K p_{i,t}v_{i,t}^2}{K(1-\gamma)\max_{i,t} p_{i,t}v_{i,t}}\geq\frac{\gamma}{K\max_{i,t} p_{i,t}v_{i,t}}\mathbb{E}_{i\sim q_t} v_{i,t}^2\geq\eta\Theta(\eta B)\mathbb{E}_{i\sim q_t} v_{i,t}^2,$$

where the last inequality uses (11). We have thus proved

$$\sum_{t=1}^n\sum_{i=1}^K p_{i,t}v_{i,t}\geq\frac{1-\gamma}{\eta}\log\mathbb{E}_{i\sim p_1}\exp(\eta V_{i,n})\geq(1-\gamma)\left(\max_{1\leq i\leq K} V_{i,n}-\frac{\log K}{\eta}\right),$$

hence the announced result.

C.3 Recovering Theorem 3 from Theorem 27

We start with straightforward computations to bound the first sum in (25). We have $\psi^{-1}(x)=\frac{1}{\eta}\log(x-\gamma/K)$ which admits as a primitive $\int\psi^{-1}(u)du=\frac{1}{\eta}\left[(u-\gamma/K)\log(u-\gamma/K)-u\right]$. Thus one immediately gets

$$\begin{aligned} &-\int_{p_{i,n+1}}^{1/K}\psi^{-1}(u)du-p_{i,n+1}\psi^{-1}(p_{i,n+1})\\ &\qquad=\frac{1}{\eta}\left(\frac{1}{K}-\frac{1-\gamma}{K}\log\left(\frac{1-\gamma}{K}\right)-p_{i,n+1}-\frac{\gamma}{K}\log\left(p_{i,n+1}-\frac{\gamma}{K}\right)\right). \end{aligned}$$

Summing over i proves that

$$-\sum_{i=1}^K\left(p_{i,n+1}\psi^{-1}(p_{i,n+1})+\int_{p_{i,n+1}}^{1/K}\psi^{-1}(u)du\right)=\frac{1-\gamma}{\eta}\log\left(\frac{K}{1-\gamma}\right)-\frac{\gamma}{K}\sum_{i=1}^K\psi^{-1}(p_{i,n+1}).$$

With the notations of Theorem 27, we need now to bound ρ and A_t. For the former, we use Lemma 28 (page 2810) which directly shows $\rho\leq\exp(\eta B)$. For the latter we distinguish two cases. If $\gamma=0$ we use

$$A_t\leq B_t^2\sum_{i=1}^K\psi'\circ\psi^{-1}(p_{i,t})=\eta B_t^2,$$

which concludes the proof of the weakened version of (9) with $\frac{\eta}{8}$ replaced by $\frac{\eta}{2}\exp(2\eta B)$. On the other hand if $\gamma>0$ we use

$$A_t\leq(1+\rho^2)\sum_{i=1}^K\psi'\circ\psi^{-1}(p_{i,t})v_{i,t}^2\leq(1+\rho^2)\eta\sum_{i=1}^K p_{i,t}v_{i,t}^2.$$

From Theorem 27, when the weakened version of (11) holds, that is when

$$\gamma \geq K \frac{\eta \exp(2B\eta)\,[1+\exp(2B\eta)]}{2} \max_{i,t} p_{i,t} v_{i,t},$$

we have

$$\begin{aligned}
&C_n - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t} \\
&\leq \frac{1-\gamma}{\eta}\log\left(\frac{K}{1-\gamma}\right) - \frac{\gamma}{K}\sum_{t=1}^{K}\left(\sum_{t=1}^{n} v_{i,t} - C_n\right) + \frac{\eta\exp(2B\eta)\,[1+\exp(2B\eta)]}{2}\sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t}^2 \\
&\leq \frac{1-\gamma}{\eta}\log\left(\frac{K}{1-\gamma}\right) + \gamma C_n,
\end{aligned}$$

hence

$$(1-\gamma)\left(C_n + \frac{\log(1-\gamma)}{\eta}\right) - \sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t} v_{i,t} \leq (1-\gamma)\frac{\log K}{\eta}.$$

This gives the desired result since we have

$$C_n + \frac{\log(1-\gamma)}{\eta} = \frac{1}{\eta}\log\left(\sum_{j=1}^{K}\exp(\eta V_{i,n})\right) \geq \max_{1\leq i\leq K} V_{i,n}.$$

C.4 Proof of Theorem 8 (page 2797)

We will use the following version of Bernstein's inequality for martingales.

Theorem 31 *Let $\mathcal{F}_1 \subset \cdots \subset \mathcal{F}_n$ be a filtration, and $X_1, \ldots, X_n$ random variables such that $|X_t| \leq b$ for some $b > 0$, X_t is $\mathcal{F}_t$-measurable, $\mathbb{E}(X_t|\mathcal{F}_{t-1}) = 0$ and $\mathbb{E}(X_t^2|\mathcal{F}_{t-1}) \leq v$ for some $v > 0$. Then, for any $t > 0$, we have*

$$\mathbb{P}\Big(\sum_{t=1}^{n} X_t \geq t\Big) \leq \exp\left(-\frac{t^2}{2nv + 2bt/3}\right), \tag{34}$$

and for any $\delta > 0$, with probability at least $1-\delta$, we have

$$\sum_{t=1}^{n} X_t \leq \sqrt{2nv\log(\delta^{-1})} + \frac{b\log(\delta^{-1})}{3}.$$

Proof of Theorem 31 Both inequalities come from Result (1.6) of Freedman (1975). The first inequality then uses $(1+x)\log(1+x) - x \geq \frac{x^2}{2+2x/3}$, while the other uses Inequality (45) of Audibert et al. (2009). This last inequality allows to remove the $\sqrt{2}$ factor appearing in Lemma A.8 of Cesa-Bianchi and Lugosi (2006). ■

We start the proof of Theorem 8 by noting that, since $R_n \leq n$, the result is trivial for $\delta \leq 2K\exp(-m/27)$ so that we assume hereafter that $\delta \geq 2K\exp(-m/27)$, or equivalently $\frac{\log(2K\delta^{-1})}{m} \leq \frac{1}{27}$. We consider the event $\mathcal{E}$ on which we simultaneously have

$$\sum_{t=1}^{n} Z_t \leq m, \tag{35}$$

$$-\sum_{t=1}^{n} g_{I_t,t} \leq -\sum_{t=1}^{n}\sum_{k=1}^{K} p_{k,t} g_{k,t} + \sqrt{\frac{n\log(4\delta^{-1})}{2}}, \tag{36}$$

and

$$\max_{1\leq i\leq K} \sum_{t=1}^{n} \left(g_{i,t} - \frac{\eta}{8\varepsilon} - \sum_{k=1}^{K} p_{k,t} g_{k,t}\right)\left(1 - \frac{Z_t}{\varepsilon}\right) \leq 2\sqrt{\frac{n\log(2K\delta^{-1})}{\varepsilon}} + \frac{\log(2K\delta^{-1})}{2\varepsilon}. \tag{37}$$

Let us first prove that this event holds with probability at least $1-\delta$. From (34), we have

$$\mathbb{P}\left(\sum_{t=1}^{n} Z_t > m\right) \leq \exp\left(-\frac{m^2/16}{3m/2+m/6}\right) \leq \exp\left(-\frac{m}{27}\right) \leq \frac{\delta}{4}$$

So (35) holds with probability at least $1-\delta/4$. From the concentration of martingales with bounded differences (Hoeffding, 1963; Azuma, 1967), (36) holds with probability at least $1-\delta/4$. For $\eta/(8\varepsilon) \leq \sqrt{2}-1$ (which is true for our particular η), we can apply Theorem 31 with $b = \sqrt{2}/\varepsilon$ and $v = 2/\varepsilon$ to the random variables $\left(g_{i,t} - \frac{\eta}{8\varepsilon} - \sum_{k=1}^{K} p_{k,t} g_{k,t}\right)\left(1 - \frac{Z_t}{\varepsilon}\right)$. We get that for a fixed $i \in \{1,\dots,K\}$, with probability at least $1-\delta/(2K)$, we have

$$\sum_{t=1}^{n} \left(g_{i,t} - \frac{\eta}{8\varepsilon} - \sum_{k=1}^{K} p_{k,t} g_{k,t}\right)\left(1 - \frac{Z_t}{\varepsilon}\right) \leq 2\sqrt{\frac{n\log(2K\delta^{-1})}{\varepsilon}} + \frac{\log(2K\delta^{-1})}{2\varepsilon}.$$

From a union bound, we get that (37) holds with probability at least $1-\delta/2$. Using again a union bound, we thus have proved that the event $\mathcal{E}$ holds with probability at least $1-\delta$.

Now, on the event $\mathcal{E}$, by combining (36) and (37), we obtain

$$\begin{aligned}
R_n = \max_{1\leq i\leq K} \sum_{t=1}^{n} g_{i,t} - \sum_{t=1}^{n} g_{I_t,t} &\leq \sqrt{\frac{n\log(4\delta^{-1})}{2}} + \max_{1\leq i\leq K} \sum_{t=1}^{n} g_{i,t} - \sum_{t=1}^{n}\sum_{k=1}^{K} p_{k,t} g_{k,t} \\
&\leq \sqrt{\frac{n\log(4\delta^{-1})}{2}} + \max_{1\leq i\leq K} \sum_{t=1}^{n} g_{i,t}\frac{Z_t}{\varepsilon} - \sum_{t=1}^{n}\sum_{k=1}^{K} p_{k,t} g_{k,t}\frac{Z_t}{\varepsilon} \\
&\quad + \sum_{t=1}^{n} \frac{\eta}{8\varepsilon}\left(1 - \frac{Z_t}{\varepsilon}\right) + 2\sqrt{\frac{n\log(2K\delta^{-1})}{\varepsilon}} + \frac{\log(2K\delta^{-1})}{2\varepsilon}
\end{aligned}$$

Since we have $\sum_{t=1}^{n} Z_t \leq m$, the rewards received by the forecaster are equal to the rewards which would receive the forecaster that uses Z_t to decide whether he asks for the gains or not, whatever $\sum_{s=1}^{t-1} Z_s$ is. This enables us to use (9) (which holds with probability one). We obtain

$$\begin{aligned}
R_n &\leq \sqrt{\frac{n\log(4\delta^{-1})}{2}} + \frac{\log K}{\eta} + \frac{n\eta}{8\varepsilon} + 2\sqrt{\frac{n\log(2K\delta^{-1})}{\varepsilon}} + \frac{\log(2K\delta^{-1})}{2\varepsilon} \\
&\leq \sqrt{\frac{n\log(4\delta^{-1})}{2}} + \frac{\log K}{\eta} + \frac{n^2\eta}{6m} + 4n\sqrt{\frac{\log(2K\delta^{-1})}{3m}} + \frac{2n\log(2K\delta^{-1})}{3m}.
\end{aligned}$$

From the inequalities $m \leq n$, $K \geq 2$ and $\frac{\log(2K\delta^{-1})}{m} \leq \frac{1}{27}$, this implies

$$R_n \leq \frac{10n}{3}\sqrt{\frac{\log(2K\delta^{-1})}{m}} + \frac{\log K}{\eta} + \frac{\eta n^2}{6m}.$$

The first inequality of the theorem is then obtained by plugging $\eta = \frac{\sqrt{m\log K}}{n}$. The second inequality is derived by integrating the deviations using the standard formula $\mathbb{E}W \leq \int_0^1 \frac{1}{\delta}\mathbb{P}(W > \log(\delta^{-1}))d\delta$.

C.5 Proof of Theorem 9 (page 2797)

The proof goes exactly like for Theorem 8. We just use (6) instead of (9).

C.6 Proof of Theorem 12 (page 2799)

The bound is trivial for $9K\log(3K) \geq n$. So we consider hereafter that $\gamma = 2\sqrt{\frac{K\log(3K)}{n}} < \frac{2}{3}$. The result is then a direct consequence of Theorem 21 with $G_0 = n$. The inequality on $\mathbb{E}R_n$ comes by integrating the deviations.

C.7 Proof of Theorem 13 (page 2800)

First note that (16) holds for $9\sqrt{nK} \geq n$ since we trivially have $R_n \leq n$. For $9\sqrt{nK} < n$, we apply (18) with $G_0 = n > 81K$, and obtain $R_n \leq 8.5\sqrt{nK} + \sqrt{2nK}\log(\delta^{-1})$. This implies (16) and also (17) by using Proposition 33.

For the last assertions, we proceed similarly. They trivially hold for $9\sqrt{nK\log(3K)} \geq n$. For $n > 9\sqrt{nK\log(3K)}$, we apply Theorem 20 with $G_0 = n$, and obtain

$$R_n \leq \frac{9}{2}\sqrt{nK\log(3K)} + 4\sqrt{\frac{nK}{\log(3K)}} + \sqrt{\frac{2nK}{\log(3K)}}\log(K\delta^{-1}).$$

By using $\frac{1}{\sqrt{\log(3K)}} \leq \frac{1}{\log(6)}\sqrt{\log(3K)}$, this independently implies

$$R_n \leq 9\sqrt{nK\log(3K)} + \sqrt{\frac{2nK}{\log(3K)}}\log(\delta^{-1}),$$

and by integration,

$$\mathbb{E}R_n \leq 9\sqrt{nK\log(3K)},$$

hence the desired inequalities.

C.8 Proof of Theorem 17 (page 2802)

The proof follows the scheme described in Section 5.1.2. In particular let

$$\nu = \frac{\gamma\varepsilon}{\beta K} + \frac{1}{\log\left(1 - \frac{\beta K}{\gamma\varepsilon}\right)}.$$

Then we have

$$\begin{aligned}
-\sum_{i=1}^{K} p_{i,t} v_{i,t} &= Z_t \frac{p_{I_t,t}}{\beta}\log\left(1 - \frac{\beta g_{I_t,t}}{\varepsilon p_{I_t,t}}\right) \\
&\geq Z_t\left(-\frac{g_{I_t,t}}{\varepsilon} + \frac{\nu g_{I_t,t}}{\varepsilon}\log\left(1 - \frac{\beta g_{I_t,t}}{\varepsilon p_{I_t,t}}\right)\right) \\
&\geq -g_{I_t,t}\frac{Z_t}{\varepsilon} - \frac{\nu\beta}{\varepsilon}\sum_{i=1}^{K} v_{i,t}.
\end{aligned}$$

Moreover with a simple application of Theorem 31 we have that with probability at least $1-\delta$,

$$-\sum_{t=1}^{n} g_{I_t,t} \leq -\sum_{t=1}^{n} g_{I_t,t}\frac{Z_t}{\varepsilon} + \sqrt{\frac{2n\log\delta^{-1}}{\varepsilon}} + \frac{\log\delta^{-1}}{3\varepsilon}.$$

Now note that the sequence $W_t = \exp\big(\beta G_{i,t} - \beta V_{i,t}\big)$, $t = 1,\dots,n$, is a supermartingale over the filtration generated by $(g_t, I_t, Z_t), t = 1,\dots,n$. Indeed, we have for any $t \in \{1,\dots,n\}$,

$$\mathbb{E}_{I_t\sim p_t, Z_t} \exp(-\beta v_{i,t}) = 1-\varepsilon+\varepsilon\left(1-\frac{\beta g_{i,t}}{\varepsilon}\right) = 1-\beta g_{i,t} \leq \exp(-\beta g_{i,t}).$$

Thus, with probability at least $1-\delta$, we have against deterministic adversaries

$$\max_{1\leq i\leq K} V_{i,n} \geq G_{\max} - \frac{\log\delta^{-1}}{\beta},$$

and against general adversaries

$$\max_{1\leq i\leq K} V_{i,n} \geq G_{\max} - \frac{\log(K\delta^{-1})}{\beta}.$$

Now we apply (8) of Theorem 2. Let $c = -\frac{\gamma}{\beta K}\log\big(1-\frac{\beta K}{\gamma\varepsilon}\big)$. If $c < q\eta\big(\frac{\gamma}{(q-1)K}\big)^{(q-1)/q}$ then we have

$$(1-\gamma)\left(\max_{1\leq i\leq K} V_{i,n}\right) - (1+\gamma\zeta)\sum_{t=1}^{n}\sum_{i=1}^{K} p_{i,t}v_{i,t} \leq \frac{q}{q-1}\eta K^{\frac{1}{q}},$$

where

$$\zeta = \frac{1}{(q-1)K}\Big(\frac{(q-1)cK\mu(1+\mu)}{2\gamma\eta}\Big)^q,$$

with

$$\mu = \exp\left\{\frac{2(q+1)c}{\eta}\left(\frac{K}{\gamma}\right)^{(q-1)/q}\left(1-\frac{c}{q\eta}\left(\frac{(q-1)K}{\gamma}\right)^{(q-1)/q}\right)^{-q}\right\}.$$

Thus, in the case of a deterministic adversaries, we obtain

$$\begin{aligned}
&G_{\max} - \sum_{t=1}^{n} g_{I_t,t} \\
&\leq \big(\gamma(1+\zeta) + \frac{\nu\beta K}{\varepsilon}\big)n + \frac{q}{q-1}\eta K^{\frac{1}{q}} + \frac{\log(2\delta^{-1})}{\beta} + n\sqrt{\frac{2\log(2\delta^{-1})}{m}} + n\frac{\log(2\delta^{-1})}{3m} \\
&= \frac{n}{m}\left(\big(\gamma(1+\zeta)+\nu\beta' K\big)m + \frac{q}{q-1}\eta' K^{\frac{1}{q}} + \frac{\log(2\delta^{-1})}{\beta'}\right) + n\sqrt{\frac{2\log(2\delta^{-1})}{m}} + n\frac{\log(2\delta^{-1})}{3m}, \quad (38)
\end{aligned}$$

where $\beta' = \beta/\varepsilon$ and $\eta' = \eta\varepsilon$. One can see that the term into parenthesis in (38) is exactly the same than the right hand side of (43), up to the relabelling of β and η into β' and η'. This allows us to use the same numerical application as in Section C.9 (up to the additional terms outside of the parenthesis in (43)). One can apply the same technique in the case of a general adversary.

C.9 Proof of Theorem 18, Corollary 19 and Theorem 20 (page 2803)

Consider parameters $q > 1$, $0 < \gamma < 1$, $\eta > 0$ and $\beta > 0$ such that $\beta K < \gamma$ and such that the real number $c = -\frac{\gamma}{\beta K}\log\left(1 - \frac{\beta K}{\gamma}\right)$ satisfies $c < q\eta\left(\frac{\gamma}{(q-1)K}\right)^{(q-1)/q}$. From (8) of Theorem 2, since we have $v_{i,t} = \frac{c_t}{p_{i,t}}\mathbb{I}_{I_t=i}$ with $c_t = -\frac{p_{i,t}}{\beta}\log\left(1 - \frac{\beta g_{i,t}}{p_{i,t}}\right) \le -\frac{\gamma}{\beta K}\log\left(1 - \frac{\beta K}{\gamma}\right) = c$, we have

$$(1-\gamma)\left(\max_{1\le i\le K} V_{i,n}\right) + \frac{1+\gamma\zeta}{\beta}\sum_{t=1}^{n} p_{I_t,t}\log\left(1 - \frac{\beta g_{I_t,t}}{p_{I_t,t}}\right) \le \frac{q}{q-1}\eta K^{\frac{1}{q}}, \tag{39}$$

where

$$\zeta = \frac{1}{(q-1)K}\left(\frac{(q-1)cK\mu(1+\mu)}{2\gamma\eta}\right)^{q},$$

with

$$\mu = \exp\left\{\frac{2(q+1)c}{\eta}\left(\frac{K}{\gamma}\right)^{(q-1)/q}\left(1 - \frac{c}{q\eta}\left(\frac{(q-1)K}{\gamma}\right)^{(q-1)/q}\right)^{-q}\right\}.$$

Let

$$\nu = \frac{\gamma}{\beta K} + \frac{1}{\log(1-\beta K/\gamma)}.$$

The function $x \mapsto \frac{1}{x} + \frac{1}{\log(1-x)}$ is increasing on $(0, +\infty)$. So we have

$$\frac{1}{\log(1 - \beta g_{I_t,t}/p_{I_t,t})} + \frac{p_{I_t,t}}{\beta g_{I_t,t}} \le \nu,$$

hence

$$\frac{p_{I_t,t}}{\beta}\log\left(1 - \frac{\beta g_{I_t,t}}{p_{I_t,t}}\right) \ge -g_{I_t,t} + \nu g_{I_t,t}\log\left(1 - \frac{\beta g_{I_t,t}}{p_{I_t,t}}\right) \ge -g_{I_t,t} - \nu\beta\sum_{i=1}^{K} v_{i,t}. \tag{40}$$

Inequality (39) thus implies

$$(1-\gamma)\left(\max_{1\le i\le K} V_{i,n}\right) - (1+\gamma\zeta)\sum_{t=1}^{n} g_{I_t,t} - (1+\gamma\zeta)\nu\beta\sum_{i=1}^{K} V_{i,n} \le \frac{q}{q-1}\eta K^{\frac{1}{q}},$$

which leads to

$$\left(1 - \gamma - (1+\gamma\zeta)\nu\beta K\right)\left(\max_{1\le i\le K} V_{i,n}\right) - (1+\gamma\zeta)\sum_{t=1}^{n} g_{I_t,t} \le \frac{q}{q-1}\eta K^{\frac{1}{q}}, \tag{41}$$

We now provide a high probability lower bound of the left-hand side. The technical tool (essentially deviation inequalities for supermartingales) comes from Section 6.8 of Cesa-Bianchi and Lugosi (2006).

For any $t \in \{1, \dots, n\}$, we have

$$\mathbb{E}_{I_t\sim p_t}\exp(-\beta v_{i,t}) = \mathbb{E}_{I_t\sim p_t}\exp\left\{\log\left(1 - \frac{\beta g_{i,t}}{p_{i,t}}\right)\mathbb{I}_{I_t=i}\right\} = 1 - \beta g_{i,t} \le \exp(-\beta g_{i,t}).$$

This implies that the sequence $W_t = \exp\left(\beta G_{i,t} - \beta V_{i,t}\right)$, $t = 1, \dots, n$, forms a supermartingale over the filtration generated by (g_t, I_t), $t = 1, \dots, n$. Indeed, we have

$$\mathbb{E}\left(\exp(W_t)|(g_s, I_s), s = 1, \dots, t-1\right) = \mathbb{E}_{g_t|(g_s,I_s),s=1,\dots,t-1}\mathbb{E}_{I_t\sim p_t}\exp(W_t) \le \exp(W_{t-1}).$$

So, we have $\mathbb{E}\exp(W_n) \leq \mathbb{E}\exp(W_1) \leq 1$, which implies that with probability at least $1-\delta$, $V_{i,n} \geq G_{i,n} - \frac{\log(\delta^{-1})}{\beta}$ with probability at least $1-\delta$. So, for any fixed $k \in \{1,\dots,K\}$, we have

$$\max_{1\leq i\leq K} V_{i,n} \geq G_{k,n} - \frac{\log(\delta^{-1})}{\beta}. \tag{42}$$

Combining (41) and (42), we obtain that for any $\delta > 0$ and any fixed $k \in \{1,\dots,K\}$, with probability at least $1-\delta$, we have

$$\big(1-\gamma-(1+\gamma\zeta)\nu\beta K\big)G_{k,n} - (1+\gamma\zeta)\sum_{t=1}^{n} g_{I_t,t} \leq \frac{q}{q-1}\eta K^{\frac{1}{q}} + \frac{\log(\delta^{-1})}{\beta},$$

hence

$$G_{k,n} - \sum_{t=1}^{n} g_{I_t,t} \leq \big(\gamma(1+\zeta)+\nu\beta K\big)G_{k,n} + \frac{q}{q-1}\eta K^{\frac{1}{q}} + \frac{\log(\delta^{-1})}{\beta}. \tag{43}$$

Now, for $G_0 \geq 81K$, let us take

$$q = 2, \quad \gamma = 3\sqrt{\frac{K}{G_0}}, \quad \beta = \frac{1}{\sqrt{2KG_0}}, \quad \text{and} \quad \eta = 2\sqrt{G_0}.$$

Then we have $c \approx 1.14$, $\nu \approx 0.522$, $\mu \leq 2.09$, $\zeta \leq 0.377$, and

$$G_{k,n} - \sum_{t=1}^{n} g_{I_t,t} \leq 4.5\sqrt{K\frac{G_{\max}^2}{G_0}} + 4\sqrt{KG_0} + \sqrt{2KG_0}\log(\delta^{-1}). \tag{44}$$

For deterministic adversaries, the arm achieving a cumulative reward equal to $G_{\max}$ is deterministic (it does not depend on the randomization of the forecaster). So, we may take k equal to this fixed arm, and obtain (18).

To prove the inequality for a fully oblivious adversary, let us take $k \in \operatorname{argmax}_{i\in\{1,\dots,K\}} \mathbb{E}G_{i,n}$. From (44) which holds with probability at least $1-\delta$, it suffices to prove that with probability at least $1-\delta$, we have $G_{k,n} \geq G_{\max} - \sqrt{8\log(K\delta^{-1})G_{\max}}$. Let $\lambda > 0$. Since the reward vectors $g_1,\dots,g_n$ are independent, and from the inequality $\exp(\lambda x) \leq 1 + [\exp(\lambda)-1]x$ for any $x \in [0,1]$ (by convexity of the exponential function), for any $j \neq k$, from Lemma 35, we have

$$\mathbb{E}\exp(\lambda G_{j,n}) = \prod_{t=1}^{n}\mathbb{E}\exp(\lambda g_{j,t}) \leq \prod_{t=1}^{n}\exp\big[\big(\exp(\lambda)-1\big)\mathbb{E}g_{j,t}\big] = \exp\big[\big(\exp(\lambda)-1\big)\mathbb{E}G_{j,n}\big],$$

and

$$\begin{aligned}
\mathbb{E}\exp(-\lambda G_{k,n}) &= \prod_{t=1}^{n}\mathbb{E}\exp(-\lambda g_{k,t}) \\
&\leq \prod_{t=1}^{n}\mathbb{E}\left(1-\lambda g_{k,t}+\frac{1}{2}\lambda^2 g_{k,t}^2\right) \\
&\leq \prod_{t=1}^{n}\left(1-\lambda\Big(1-\frac{\lambda}{2}\Big)\mathbb{E}g_{k,t}\right) \\
&\leq \prod_{t=1}^{n}\exp\left[-\lambda\Big(1-\frac{\lambda}{2}\Big)\mathbb{E}g_{k,t}\right] = \exp\left[-\lambda\Big(1-\frac{\lambda}{2}\Big)\mathbb{E}G_{k,n}\right].
\end{aligned}$$

This implies respectively that for any $j \neq k$, with probability at least $1-\delta$,

$$G_{j,n} \leq \mathbb{E}G_{j,n} + \lambda\Theta(\lambda)\mathbb{E}G_{j,n} + \frac{\log(\delta^{-1})}{\lambda},$$

and with probability at least $1-\delta$,

$$\mathbb{E}G_{k,n} \leq G_{k,n} + \frac{\lambda}{2}\mathbb{E}G_{k,n} + \frac{\log(\delta^{-1})}{\lambda}.$$

By optimizing the free parameter λ (using Lemma 32 below), and from a union bound, with probability at least $1-\delta$, we simultaneously have

$$G_{j,n} \leq \mathbb{E}G_{j,n} + \sqrt{2\mathbb{E}G_{j,n}\log(K\delta^{-1})} + \frac{\log(K\delta^{-1})}{3},$$

and

$$\mathbb{E}G_{k,n} \leq G_{k,n} + \sqrt{2\mathbb{E}G_{k,n}\log(K\delta^{-1})}.$$

Since we have $\mathbb{E}G_{k,n} \geq \mathbb{E}G_{j,n}$, we get consecutively $G_{k,n} \geq \mathbb{E}G_{j,n} - \sqrt{2\mathbb{E}G_{j,n}\log(K\delta^{-1})}$, and after computations, $G_{k,n} \geq G_{j,n} - \sqrt{8G_{j,n}\log(K\delta^{-1})}$ for any $j \neq k$. With probability at least $1-\delta$, we thus have $G_{k,n} \geq G_{\max} - \sqrt{8\log(K\delta^{-1})G_{\max}}$, which concludes the proof of (19).

To prove Corollary 19, first note that (20) holds for $9\sqrt{KG_{\max}} \geq G_{\max}$ since we trivially have $R_n \leq G_{\max}$. For $9\sqrt{KG_{\max}} < G_{\max}$, we may apply Theorem 18 with $G_0 = G_{\max}$ since $G_{\max} > 81K$, and obtain $R_n \leq 8.5\sqrt{KG_{\max}} + \sqrt{2KG_{\max}}\log(\delta^{-1})$. This implies (20) and (21).

Lemma 32 *Let* $\Theta(\lambda) = \frac{\exp(\lambda)-1-\lambda}{\lambda^2}$. *For any* $A > 0$, $\inf_{\lambda>0}\left\{\lambda\Theta(\lambda) + \frac{A^2}{2\lambda}\right\} \leq A + A^2/6$.

Proof Considering $\lambda = \log(1+A)$, we have $\inf_{\lambda>0}\left\{\lambda\Theta(\lambda) + \frac{A^2}{2\lambda}\right\} \leq \log(1+A)\Theta[\log(1+A)] + \frac{A^2}{2\log(1+A)} = A + \frac{A^2}{6} - \frac{1+A+\frac{A^2}{6}}{\log(1+A)}\Phi(A)$ where $\Phi(A) \triangleq \log(1+A) - \frac{A+\frac{A^2}{2}}{1+A+\frac{A^2}{6}}$. Since $\Phi(0) = 0$ and $\Phi'(A) = \frac{A^4}{36(1+A)(1+A+A^2/6)^2} \geq 0$, we get $\Phi(A) \geq 0$, hence the result. ■

To prove Theorem 20, we replace (42), which holds with probability at least $1-\delta$, by

$$\max_{1\leq i\leq K} V_{i,n} \geq G_{\max} - \frac{\log(\delta^{-1})}{\beta},$$

which, by a union bound, holds with probability at least $1-K\delta$. (It is this union bound that makes the $\log K$ factor appears in the bound.) This leads to the following modified version of (43): with probability $1-K\delta$,

$$G_{\max} - \sum_{t=1}^{n} g_{I_t,t} \leq \big(\gamma(1+\zeta) + \nu\beta K\big)G_{\max} + \frac{q}{q-1}\eta K^{\frac{1}{q}} + \frac{\log(\delta^{-1})}{\beta}.$$

Now, for $G_0 \geq 81K\log(3K)$, let us take $q = 2$,

$$\gamma = 3\sqrt{\frac{K\log(3K)}{G_0}}, \quad \beta = \sqrt{\frac{\log(3K)}{2KG_0}}, \quad \text{and} \quad \eta = 2\sqrt{\frac{G_0}{\log(3K)}}.$$

Then we have $c \approx 1.14, \nu \approx 0.522, \mu \leq 2.09, \zeta \leq 0.377$, and

$$G_{\max} - \sum_{t=1}^{n} g_{I_t,t} \leq 4.5\sqrt{\frac{G_{\max}^2}{G_0} K \log(3K)} + 4\sqrt{\frac{KG_0}{\log(3K)}} + \sqrt{\frac{2KG_0}{\log(3K)}} \log(\delta^{-1}).$$

This inequality holds with probability at least $1 - K\delta$. This implies the result of Theorem 20.

C.10 Proof of Theorem 21 (page 2803)

Consider parameters $q > 1, 0 < \gamma < 1, \eta > 0$ and $\beta > 0$ such that $\beta K < \gamma$. Introduce $c = -\frac{\gamma}{\beta K} \log\left(1 - \frac{\beta K}{\gamma}\right)$. We have $\max_{i,t} p_{i,t} v_{i,t} \leq c$. So (11) holds as soon as

$$\gamma \geq K\eta c \Theta\left(-\frac{\eta}{\beta} \log\left(1 - \frac{\beta K}{\gamma}\right)\right). \tag{45}$$

From (12) and as in the proof of Theorem 13, by using (40) with $\nu = \frac{\gamma}{\beta K} + \frac{1}{\log(1-\beta K/\gamma)}$, we obtain

$$(1-\gamma)\left(\max_{1 \leq i \leq K} V_{i,n}\right) - \sum_{t=1}^{n} g_{I_t,t} - \nu\beta \sum_{i=1}^{K} V_{i,n} \leq (1-\gamma)\frac{\log K}{\eta},$$

hence

$$(1 - \gamma - \nu\beta K)\left(\max_{1 \leq i \leq K} V_{i,t}\right) - \sum_{t=1}^{n} g_{I_t,t} \leq (1-\gamma)\frac{\log K}{\eta}.$$

From the same argument as in the proof of Theorem 18, for any $i \in \{1, \ldots, K\}$, we have $\mathbb{E}\exp\left(\beta G_{i,n} - \beta V_{i,n}\right) \leq 1$, and for any $\delta > 0$, $V_{i,n} \geq G_{i,n} - \frac{\log(\delta^{-1})}{\beta}$ holds with probability at least $1 - \delta$. By a union bound, we get that with probability at least $1 - K\delta$,

$$\max_{1 \leq i \leq K} V_{i,n} \geq G_{\max} - \frac{\log(\delta^{-1})}{\beta}.$$

With probability at least $1 - \delta$, we thus have

$$G_{\max} - \sum_{t=1}^{n} g_{I_t,t} \leq (\gamma + \nu\beta K) G_{\max} + \frac{\log(K\delta^{-1})}{\beta} + \frac{\log K}{\eta}.$$

This inequality holds for any parameters $q > 1, 0 < \gamma < 1, \eta > 0$ and $\beta > 0$ such that the inequality $\beta K < \gamma$ and (45) hold. We choose

$$\gamma = 2\sqrt{\frac{K\log(3K)}{G_0}}, \quad \beta = \sqrt{\frac{\log(3K)}{2KG_0}}, \quad \text{and} \quad \eta = 2\sqrt{\frac{\log(3K)}{KG_0}},$$

which gives $c \approx 1.23, \nu \approx 0.536$, and

$$G_{\max} - \sum_{t=1}^{n} g_{I_t,t} \leq \frac{5}{2}\sqrt{\frac{G_{\max}^2}{G_0} K \log(3K)} + \sqrt{\frac{2KG_0}{\log(3K)}} \log(K\delta^{-1}) + \frac{1}{2}\sqrt{KG_0 \log(3K)}.$$

C.11 Proof of Theorem 22 (page 2804)

Consider parameters $0 < \gamma < 1$, $\eta > 0$ and $\beta > 0$. We have $\max_{i,t} p_{i,t} v_{i,t} \leq 1 + \frac{\beta K}{\gamma}$ and $\max_{i,t} v_{i,t} \leq (1+\beta)\frac{K}{\gamma}$. So (11) holds as soon as

$$\gamma \geq K\eta\Big(1 + \frac{\beta K}{\gamma}\Big)\Theta\bigg(\frac{\eta(1+\beta)K}{\gamma}\bigg). \tag{46}$$

Then, from (12), we have

$$(1-\gamma)\bigg(\max_{1\leq i\leq K} V_{i,n}\bigg) - \sum_{t=1}^{n} g_{I_t,t} - \beta nK \leq (1-\gamma)\frac{\log K}{\eta}.$$

Let $\xi_t = \max_{1\leq i\leq K} V_{i,t} - \min_{1\leq j\leq K} V_{j,t}$ and $\xi = \max_{1\leq t\leq n} \xi_t$. Consider a fixed switching strategy $(i_1,\ldots,i_n) \in \{1,\ldots,K\}^n$, and let $V_{(i_1,\ldots,i_n)} = \sum_{t=1}^{n} v_{i_t,t}$. One can easily check that $\max\limits_{1\leq i\leq K} V_{i,n} \geq V_{(i_1,\ldots,i_n)} - \xi\mathcal{S}(i_1,\ldots,i_n)$, and consequently

$$\max_{1\leq i\leq K} V_{i,n} \geq \max_{(i_1,\ldots,i_n):\mathcal{S}(i_1,\ldots,i_n)\leq S} V_{(i_1,\ldots,i_n)} - \xi S.$$

Since $\exp(-x) \leq 1 - x + x^2/2$ for $x \leq 0$, we have for any $t \in \{1,\ldots,n\}$ and any $i \in \{1,\ldots,K\}$

$$\begin{aligned}
\mathbb{E}_{I_t\sim p_t} \exp\bigg(-2\beta g_{i,t}\frac{\mathbb{I}_{I_t=i}}{p_{i,t}}\bigg) &\leq \mathbb{E}_{I_t\sim p_t}\bigg(1 - 2\beta g_{i,t}\frac{\mathbb{I}_{I_t=i}}{p_{i,t}} + 2\beta^2 g_{i,t}^2\frac{\mathbb{I}_{I_t=i}}{p_{i,t}^2}\bigg) \\
&= 1 - 2\beta g_{i,t} + 2\beta^2\frac{g_{i,t}^2}{p_{i,t}} \\
&\leq 1 - 2\beta\Big(g_{i,t} - \frac{\beta}{p_{i,t}}\Big) \\
&\leq \exp\bigg(-2\beta\Big(g_{i,t} - \frac{\beta}{p_{i,t}}\Big)\bigg),
\end{aligned}$$

hence

$$\mathbb{E}_{I_t\sim p_t} \exp\big(2\beta(g_{i,t} - v_{i,t})\big) \leq 1.$$

For a fixed $(i_1,\ldots,i_n)$, by using this inequality n times corresponding to the n time steps and their associated actions, this implies $\mathbb{E}\exp\big(2\beta(G_{(i_1,\ldots,i_n)} - V_{(i_1,\ldots,i_n)})\big) \leq 1$, hence with probability at least $1-\delta$,

$$G_{(i_1,\ldots,i_n)} - V_{(i_1,\ldots,i_n)} \leq \frac{\log(\delta^{-1})}{2\beta}.$$

Let $M = \sum_{j=0}^{S}\binom{n-1}{j}K(K-1)^j$ be the number of switching strategies of size not larger than S. By a union bound, we get that with probability at least $1-\delta$,

$$\max_{(i_1,\ldots,i_n):\mathcal{S}(i_1,\ldots,i_n)\leq S} V_{(i_1,\ldots,i_n)} \geq \max_{(i_1,\ldots,i_n):\mathcal{S}(i_1,\ldots,i_n)\leq S} G_{(i_1,\ldots,i_n)} - \frac{\log(M\delta^{-1})}{2\beta}.$$

By putting the previous inequalities together, we obtain that with probability at least $1-\delta$,

$$(1-\gamma)\max_{(i_1,\dots,i_n):\mathcal{S}(i_1,\dots,i_n)\leq S} G_{(i_1,\dots,i_n)} - \sum_{t=1}^{n} g_{I_t,t} \leq \beta nK + (1-\gamma)\frac{\log K}{\eta} + \xi S + \frac{\log(M\delta^{-1})}{2\beta},$$

hence

$$R_n^S = \max_{(i_1,\dots,i_n):\mathcal{S}(i_1,\dots,i_n)\leq S} G_{(i_1,\dots,i_n)} - \sum_{t=1}^{n} g_{I_t,t} \leq (\gamma+\beta K)n + \frac{\log K}{\eta} + \xi S + \frac{\log(M\delta^{-1})}{2\beta}.$$

We now upper bound M and ξ. We have

$$M = \sum_{j=0}^{S}\binom{n-1}{j}K(K-1)^j \leq K^{S+1}\sum_{j=0}^{S}\binom{n-1}{j} \leq K^{S+1}\left(\frac{en}{S}\right)^S = \frac{\exp(s)}{2},$$

where the second inequality comes from Sauer's lemma. Let

$$\tilde{\rho} = \exp\left((1+\beta)\frac{K\eta}{\gamma}\right)\frac{1+K\beta}{1-\gamma}.$$

By contradiction, we now prove

$$\xi \leq \tilde{\rho} - \frac{1}{\eta}\log\left(\frac{\beta}{\tilde{\rho}} - \frac{\gamma}{K}\right). \tag{47}$$

To this end, we start by bounding $C_t - C_{t-1}$. By the mean value theorem, with the notations of the third step of the proof of Theorem 27, there exists $W \in [V_{t-1}, V_t]$ such that

$$\begin{aligned}
C_t - C_{t-1} &= C(V_t) - C(V_{t-1}) \\
&= \sum_{i=1}^{K}\frac{\partial C}{\partial x_i}(W)(V_{i,t} - V_{i,t-1}) \\
&= \sum_{i=1}^{K}\frac{h_i(W)}{\sum_{j=1}^{K}h_j(W)}\frac{g_{i,t}\mathbb{I}_{I_t=i}+\beta}{f_i(V_{i,t-1})} \\
&= \frac{1}{\sum_{j=1}^{K}\eta(f_j(W)-\gamma/K)}\sum_{i=1}^{K}\eta h_i(W)\frac{g_{i,t}\mathbb{I}_{I_t=i}+\beta}{h_i(V_{i,t-1})+\eta\gamma/K} \\
&\leq \frac{1}{1-\gamma}\sum_{i=1}^{K}h_i(W)\frac{\mathbb{I}_{I_t=i}+\beta}{h_i(V_{t-1})} \leq \frac{\rho}{1-\gamma}\sum_{i=1}^{K}(\mathbb{I}_{I_t=i}+\beta) = \rho\frac{1+K\beta}{1-\gamma}.
\end{aligned}$$

From Lemma 28 (page 2810), we have $\rho \leq \exp\left((1+\beta)\frac{K\eta}{\gamma}\right)$, hence $C_t - C_{t-1} \leq \exp\left((1+\beta)\frac{K\eta}{\gamma}\right)\frac{1+K\beta}{1-\gamma} = \tilde{\rho}$. If (47) does not hold, then from Lemma 1, we have

$$\max_{1\leq t\leq n}\left(C_t - \min_{1\leq j\leq K}V_{j,t}\right) > \tilde{\rho} - \psi^{-1}(\beta/\tilde{\rho}).$$

Besides we have $C_0 - \min_{1\leq j\leq K}V_{j,0} = -\psi^{-1}(1/K) \leq \tilde{\rho} - \psi^{-1}(\beta/\tilde{\rho})$, since $K\beta \leq \tilde{\rho}$. So there exist $T \in \{1,\dots,n\}$ and $\ell \in \{1,\dots,K\}$ such that $C_{T-1} - V_{\ell,T-1} \leq \tilde{\rho} - \psi^{-1}(\beta/\tilde{\rho})$ and $C_T - V_{\ell,T} > \tilde{\rho} - \psi^{-1}(\beta/\tilde{\rho})$. In particular, we have $\psi(V_{\ell,T} - C_T + \tilde{\rho}) < \frac{\beta}{\tilde{\rho}}$, hence

$$V_{\ell,T} - V_{\ell,T-1} \geq \frac{\beta}{p_{\ell,T}} = \frac{\beta}{\psi(V_{\ell,T-1} - C_{T-1})} \geq \frac{\beta}{\psi(V_{\ell,T} - C_T + \tilde{\rho})} \geq \tilde{\rho} \geq C_T - C_{T-1},$$

which contradicts the inequality $C_{T-1} - V_{\ell,T-1} < C_T - V_{\ell,T}$. This ends the proof of (47). We have thus proved that for any $0 < \gamma < 1$, $\eta > 0$ and $\beta > 0$ such that (46) holds, we have

$$R_n^S \leq (\gamma + \beta K)n + \frac{\log K}{\eta} + S\left\{\tilde{\rho} - \frac{1}{\eta}\log\left(\frac{\beta}{\tilde{\rho}} - \frac{\gamma}{K}\right)\right\} + \frac{\log(K\delta^{-1})}{2\beta} + \frac{S\log(Ken/S)}{2\beta},$$

with $\tilde{\rho} = \frac{1+K\beta}{1-\gamma}\exp\left((1+\beta)\frac{K\eta}{\gamma}\right)$. For the numerical application, we first notice that the bound trivially holds for $7\sqrt{Ks} \geq \sqrt{n}$. For $7\sqrt{Ks} < \sqrt{n}$, with $s = S\log\left(\frac{enK}{S}\right) + 2\log K$, we choose

$$\gamma = \sqrt{\frac{Ks}{2n}}, \quad \beta = 3\sqrt{\frac{s}{nK}}, \quad \text{and} \quad \eta = \frac{1}{5}\sqrt{\frac{s}{nK}}.$$

We then use $\gamma \leq \frac{1}{7\sqrt{2}}$, $\beta K \leq \frac{3}{7}$, $\beta \leq \frac{3}{14}$ to deduce $\tilde{\rho} \leq 2.25$, and $\tilde{\rho}S \leq 0.05\sqrt{nKs}$. We check (46) by the upper bound $\frac{K\eta}{\gamma}\left(1 + \frac{\beta K}{\gamma}\right)\Theta\left(\frac{\eta(1+\beta)K}{\gamma}\right) \leq 0.84 < 1$. We also use $-\log\left(\frac{\beta}{\tilde{\rho}} - \frac{\gamma}{K}\right) \leq \frac{1}{2}\log\left(3nK/s\right) \leq \frac{1}{2}\log\left(3nK/S\right)$. We thus have

$$R_n^S \leq \left(3 + \frac{1}{\sqrt{2}} + 0.05 + \frac{5}{2} + \frac{1}{6}\right)\sqrt{nKs} + \frac{\log(\delta^{-1})}{2\beta} \leq 6.5\sqrt{nKs} + \frac{\log(\delta^{-1})}{6}\sqrt{\frac{nK}{s}}$$

The last inequality follows by integrating the deviations.

C.12 Proof of Theorem 24 (page 2806)

This proof requires some new arguments compared to the one for UCB1. First, we need to decouple the arm, while not being too loose. This is achieved by introducing appropriate stopping times. The decoupled upper bound on the pseudo-regret is (51). Secondly, we use peeling arguments to tightly control the terms in the right-hand side of (51).

We may assume $\mu_1 \geq \ldots \geq \mu_K$. Using the trivial equality $\sum_{i=1}^K \mathbb{E}T_i(n) = n$, we have

$$\begin{aligned}
\overline{R}_n &= \max_{1\leq i\leq K} \mathbb{E}\sum_{t=1}^n \left(g_{i,t} - g_{I_t,t}\right) \\
&= n\Big(\max_{1\leq i\leq K} \mathbb{E}g_{i,t}\Big) - \sum_{t=1}^n \mathbb{E}g_{I_t,t} \\
&= n\Big(\max_{1\leq i\leq K} \mu_i\Big) - \sum_{t=1}^n \mathbb{E}\mu_{I_t} \\
&= n\Big(\max_{1\leq i\leq K} \mu_i\Big) - \mathbb{E}\sum_{t=1}^n \mu_{I_t} \\
&= \left(\sum_{i=1}^K \mathbb{E}T_i(n)\right)\Big(\max_{1\leq i\leq K} \mu_i\Big) - \mathbb{E}\sum_{i=1}^K \mu_i T_i(n) = \sum_{i=1}^K \Delta_i \mathbb{E}T_i(n).
\end{aligned}$$

First step: Decoupling the arms

For an arm k_0, we trivially have $\sum_{k=1}^K \Delta_k T_k(n) \leq n\Delta_{k_0} + \sum_{k=k_0+1}^K \Delta_k T_k(n)$. Let $\Delta_{K+1} = +\infty$, $z_k = \mu_1 - \frac{\Delta_k}{2}$ for $k_0 < k \leq K+1$ and $z_{k_0} = +\infty$. Let $Z = \min_{1\leq s\leq n} B_{1,s}$ and $W_{j,k} = \mathbb{1}_{Z\in[z_{j+1},z_j)}(\Delta_k -$

$\Delta_{k_0})T_k(n)$. By using $\mathbb{E}\sum_{k=1}^{k_0} T_k(n) = n - \mathbb{E}\sum_{k=k_0+1}^{K} T_k(n)$, we get

$$\overline{R}_n = \mathbb{E}\sum_{k=1}^{K} \Delta_k T_k(n) \leq n\Delta_{k_0} + \mathbb{E}\sum_{k=k_0+1}^{K} (\Delta_k - \Delta_{k_0})T_k(n).$$

We have

$$\sum_{k=k_0+1}^{K} (\Delta_k - \Delta_{k_0})T_k(n) = \sum_{k=k_0+1}^{K}\sum_{j=k_0}^{K} W_{j,k} = \sum_{j=k_0}^{K}\sum_{k=k_0+1}^{j} W_{j,k} + \sum_{j=k_0}^{K}\sum_{k=j+1}^{K} W_{j,k}. \tag{48}$$

An Abel transformation takes care of the first sum of (48):

$$\sum_{j=k_0}^{K}\sum_{k=k_0+1}^{j} W_{j,k} \leq \sum_{j=k_0}^{K} \mathbb{1}_{Z\in[z_{j+1},z_j)} n(\Delta_j - \Delta_{k_0}) = n\sum_{j=k_0+1}^{K} \mathbb{1}_{Z<z_j}(\Delta_j - \Delta_{j-1}). \tag{49}$$

To bound the second sum of (48), we introduce the stopping times $\tau_k = \min\{t : B_{k,t} < z_k\}$ and remark that, by definition of MOSS, we have $\{Z \geq z_k\} \subset \{T_k(n) \leq \tau_k\}$, since once we have pulled τ_k times arm k its index will always be lower than the index of arm 1. This implies

$$\sum_{j=k_0}^{K}\sum_{k=j+1}^{K} W_{j,k} = \sum_{k=k_0+1}^{K}\sum_{j=k_0}^{k-1} W_{j,k} = \sum_{k=k_0+1}^{K} \mathbb{1}_{Z\geq z_k}\Delta_k T_k(n) \leq \sum_{k=k_0+1}^{K} \tau_k\Delta_k. \tag{50}$$

Combining (48), (49) and (50) and taking the expectation, we get

$$\overline{R}_n \leq n\Delta_{k_0} + \sum_{k=k_0+1}^{K} \Delta_k\mathbb{E}\tau_k + n\sum_{k=k_0+1}^{K} \mathbb{P}(Z < z_k)(\Delta_k - \Delta_{k-1}). \tag{51}$$

Let $\delta_0 = \sqrt{\frac{75K}{n}}$ and set k_0 such that $\Delta_{k_0} \leq \delta_0 < \Delta_{k_0+1}$. If $k_0 = K$, we trivially have $\overline{R}_n \leq n\delta_0 \leq \sqrt{75nK}$ so that (22) holds trivially. In the following, we thus consider $k_0 < K$.

Second step: Bounding $\mathbb{E}\tau_k$ for $k_0 + 1 \leq k \leq K$.

Let $\log_+(x) = \max(\log(x), 0)$. For $\ell_0 \in \mathbb{N}$, we have

$$\begin{aligned}
\mathbb{E}\tau_k - \ell_0 &= \sum_{\ell=0}^{+\infty} \mathbb{P}(\tau_k > \ell) - \ell_0 \\
&\leq \sum_{\ell=\ell_0}^{+\infty} \mathbb{P}(\tau_k > \ell) = \sum_{\ell=\ell_0}^{+\infty} \mathbb{P}(\forall t \leq \ell, B_{k,t} > z_k) \\
&\leq \sum_{\ell=\ell_0}^{+\infty} \mathbb{P}\left(\widehat{\mu}_{k,\ell} - \mu_k \geq \frac{\Delta_k}{2} - \sqrt{\frac{\log_+(n/K\ell)}{\ell}}\right).
\end{aligned} \tag{52}$$

Now let us take $\ell_0 = \lceil 7\log\left(\frac{n}{K}\Delta_k^2\right)/\Delta_k^2 \rceil$ with $\lceil x \rceil$ the smallest integer larger than x. For $\ell \geq \ell_0$, since $k > k_0$, we have

$$\log_+\left(\frac{n}{K\ell}\right) \leq \log_+\left(\frac{n}{K\ell_0}\right) \leq \log_+\left(\frac{n\Delta_k^2}{7K}\right) \leq \frac{\ell_0\Delta_k^2}{7} \leq \frac{\ell\Delta_k^2}{7},$$

hence $\frac{\Delta_k}{2} - \sqrt{\frac{\log_+(n/(K\ell))}{\ell}} \geq c\Delta_k$, with $c = \frac{1}{2} - \frac{1}{\sqrt{7}}$. Therefore, by using Hoeffding's inequality and (52), we get

$$\begin{aligned}\mathbb{E}\tau_k - \ell_0 &\leq \sum_{\ell=\ell_0}^{+\infty} \mathbb{P}\left(\widehat{\mu}_{k,\ell} - \mu_k \geq c\Delta_k\right) \\ &\leq \sum_{\ell=\ell_0}^{+\infty} \exp\left(-2\ell(c\Delta_k)^2\right) = \frac{\exp\left(-2\ell_0(c\Delta_k)^2\right)}{1-\exp\left(-2(c\Delta_k)^2\right)} \leq \frac{\exp(-14c^2\log(75))}{1-\exp\left(-2c^2\Delta_k^2\right)},\end{aligned}$$

where the last inequality uses $\ell_0\Delta_k^2 \geq 7\log(75)$. Plugging the value of ℓ_0, we obtain

$$\begin{aligned}\Delta_k\mathbb{E}\tau_k &\leq \Delta_k\left(1 + \frac{7\log\left(\frac{n}{K}\Delta_k^2\right)}{\Delta_k^2}\right) + \frac{\Delta_k\exp(-14c^2\log(75))}{1-\exp\left(-2c^2\Delta_k^2\right)} \\ &\leq 1 + 7\frac{\log\left(\frac{n}{K}\Delta_k^2\right)}{\Delta_k} + \frac{\exp(-14c^2\log(75))}{2c^2(1-c^2)\Delta_k},\end{aligned}$$

where the last step uses that, since $1-\exp(-x) \geq x - x^2/2$ for any $x \geq 0$, we have

$$\frac{1}{1-\exp\left(-2c^2\Delta_k^2\right)} \leq \frac{1}{2c^2\Delta_k^2 - 2c^4\Delta_k^4} \leq \frac{1}{2c^2\Delta_k^2(1-c^2)}$$

Third step: Bounding $n\sum_{k=k_0+1}^{K}\mathbb{P}(Z < z_k)(\Delta_k - \Delta_{k-1})$.

Let X_t denote the reward obtained by arm 1 when it is drawn for the t-th time. The random variables $X_1, X_2, \ldots$ are i.i.d. so that we have the maximal inequality (Hoeffding, 1963, Inequality (2.17)): for any $x > 0$ and $m \geq 1$,

$$\mathbb{P}\left(\exists s \in \{1,\ldots,m\}, \sum_{t=1}^{s}(\mu_1 - X_t) > x\right) \leq \exp\left(-\frac{2x^2}{m}\right).$$

Since $z_k = \mu_1 - \Delta_k/2$ and since $u \mapsto \mathbb{P}\left(Z < \mu_1 - u/2\right)$ is a nonincreasing function, we have

$$\sum_{k=k_0+1}^{K}\mathbb{P}(Z < z_k)(\Delta_k - \Delta_{k-1}) \leq \Delta_{k_0+1}\mathbb{P}(Z < z_{k_0+1}) + \int_{\Delta_{k_0+1}}^{1}\mathbb{P}\left(Z < \mu_1 - \frac{u}{2}\right)du. \tag{53}$$

We will now concentrate on upper bounding $\mathbb{P}\left(Z < \mu_1 - \frac{u}{2}\right)$ for a fixed $u \in [\delta_0, 1]$. Let $f(u) = 8\log\left(\sqrt{\frac{n}{K}}u\right)/u^2$. We have

$$\begin{aligned}\mathbb{P}\left(Z < \mu_1 - \frac{1}{2}u\right) &= \mathbb{P}\left(\exists 1 \leq s \leq n : \sum_{t=1}^{s}(\mu_1 - X_t) > \sqrt{s\log_+\left(\frac{n}{Ks}\right)} + \frac{su}{2}\right) \\ &\leq \mathbb{P}\left(\exists 1 \leq s \leq f(u) : \sum_{t=1}^{s}(\mu_1 - X_t) > \sqrt{s\log_+\left(\frac{n}{Ks}\right)}\right) \\ &\quad + \mathbb{P}\left(\exists f(u) < s \leq n : \sum_{t=1}^{s}(\mu_1 - X_t) > \frac{su}{2}\right).\end{aligned}$$

For the first term, we use a peeling argument with a geometric grid of the form $\frac{1}{2^{\ell+1}} f(u) \le s \le \frac{1}{2^\ell} f(u)$. The numerical constant in δ_0 ensures that $f(u) \le n/K$, which implies that for any $s \le f(u)$, $\log_+\left(\frac{n}{Ks}\right) = \log\left(\frac{n}{Ks}\right)$. We have

$$\begin{aligned}
&\mathbb{P}\left(\exists 1 \le s \le f(u) : \sum_{t=1}^{s} (\mu_1 - X_t) > \sqrt{s \log\left(\frac{n}{Ks}\right)}\right) \\
&\quad \le \sum_{\ell=0}^{+\infty} \mathbb{P}\left(\exists \frac{1}{2^{\ell+1}} f(u) \le s \le \frac{1}{2^\ell} f(u) : \sum_{t=1}^{s} (\mu_1 - X_t) > \sqrt{\frac{f(u)}{2^{\ell+1}} \log\left(\frac{n 2^\ell}{K f(u)}\right)}\right) \\
&\quad \le \sum_{\ell=0}^{+\infty} \exp\left(-2 \frac{f(u)\frac{1}{2^{\ell+1}} \log\left(\frac{n2^\ell}{Kf(u)}\right)}{f(u)\frac{1}{2^\ell}}\right) = \sum_{\ell=0}^{+\infty} \frac{Kf(u)}{n} \frac{1}{2^\ell} = \frac{16K}{nu^2} \log\left(\sqrt{\frac{n}{K}} u\right).
\end{aligned}$$

For the second term we also use a peeling argument but with a geometric grid of the form $2^\ell f(u) \le s \le 2^{\ell+1} f(u)$:

$$\begin{aligned}
&\mathbb{P}\left(\exists s \in \lceil f(u) \rceil, \dots, n\} : \sum_{t=1}^{s} (\mu_1 - X_t) > \frac{su}{2}\right) \\
&\qquad \le \sum_{\ell=0}^{+\infty} \mathbb{P}\left(\exists 2^\ell f(u) \le s \le 2^{\ell+1} f(u) : \sum_{t=1}^{s} (\mu_1 - X_t) > 2^{\ell-1} f(u) u\right) \\
&\qquad \le \sum_{\ell=0}^{+\infty} \exp\left(-2 \frac{\left(2^{\ell-1} f(u) u\right)^2}{f(u) 2^{\ell+1}}\right) \\
&\qquad = \sum_{\ell=0}^{+\infty} \exp\left(-2^\ell f(u) u^2/4\right) \\
&\qquad \le \sum_{\ell=0}^{+\infty} \exp\left(-(\ell+1) f(u) u^2/4\right) = \frac{1}{\exp\left(f(u)u^2/4\right) - 1} = \frac{1}{nu^2/K - 1}.
\end{aligned}$$

Putting together the last three computations, we obtain

$$\mathbb{P}\left(Z < \mu_1 - \frac{1}{2}u\right) \le \frac{16K}{nu^2} \log\left(\sqrt{\frac{n}{K}} u\right) + \frac{1}{nu^2/K - 1}.$$

Plugging this into (53) gives

$$\begin{aligned}
&\sum_{k=k_0+1}^{K} \mathbb{P}(Z < z_k)(\Delta_k - \Delta_{k-1}) \\
&\quad \leq \frac{16K}{n\Delta_{k_0+1}} \log\left(\sqrt{\frac{n}{K}}\Delta_{k_0+1}\right) + \frac{\Delta_{k_0+1}}{n\Delta_{k_0+1}^2/K - 1} \\
&\qquad + \left[-\frac{16K}{nu}\log\left(e\sqrt{\frac{n}{K}}u\right) + \sqrt{\frac{K}{4n}}\log\left(\frac{\sqrt{\frac{n}{K}}u-1}{\sqrt{\frac{n}{K}}u+1}\right)\right]_{\Delta_{k_0+1}}^{1} \\
&\quad \leq \frac{16K}{n\Delta_{k_0+1}} \log\left(\frac{en\Delta_{k_0+1}^2}{K}\right) + \frac{\Delta_{k_0+1}}{n\Delta_{k_0+1}^2/K - 1} + \sqrt{\frac{K}{4n}}\log\left(\frac{\sqrt{\frac{n}{K}}\Delta_{k_0+1}+1}{\sqrt{\frac{n}{K}}\Delta_{k_0+1}-1}\right) \\
&\quad \leq \frac{16K}{n\Delta_{k_0+1}} \log\left(\frac{en\Delta_{k_0+1}^2}{K}\right) + \left(\frac{75}{74} + \frac{\sqrt{75}}{\sqrt{75}-1}\right)\frac{K}{n\Delta_{k_0+1}}
\end{aligned}$$

where the penultimate inequality uses $\Delta_{k_0+1} \geq \sqrt{\frac{75K}{n}}$ and $\log(1+x) \leq x$ for any $x \geq 0$.

Gathering the results of the three steps, we get

$$\begin{aligned}
\overline{R}_n &\leq n\Delta_{k_0} + \sum_{k=k_0+1}^{K}\left(1 + 7\frac{\log\left(\frac{n}{K}\Delta_k^2\right)}{\Delta_k} + \frac{\exp(-14c^2\log(75))}{2c^2(1-c^2)\Delta_k}\right) \\
&\qquad + \frac{16K}{\Delta_{k_0+1}}\log\left(\frac{en\Delta_{k_0+1}^2}{K}\right) + \left(\frac{75}{74} + \frac{\sqrt{75}}{\sqrt{75}-1}\right)\frac{K}{\Delta_{k_0+1}} \\
&\leq n\Delta_{k_0} + K + (16+7)K\frac{\log\left(\frac{n}{K}\Delta_{k_0+1}^2\right)}{\Delta_{k_0+1}} + (16+16)\frac{K}{\Delta_{k_0+1}} \\
&\leq n\delta_0 \mathbb{I}_{\Delta \leq \delta_0} + 23K\frac{\log\left(\frac{n}{K}\Delta_{k_0+1}^2\right)}{\Delta_{k_0+1}} + \frac{33K}{\Delta_{k_0+1}} \\
&\leq 23K\frac{\log\left(\frac{n}{K}\max(\Delta,\delta_0)^2\right)}{\max(\Delta,\delta_0)} + \frac{108K}{\max(\Delta,\delta_0)} \\
&\leq 23K\frac{\log\left(\frac{110n}{K}\max(\Delta,\delta_0)^2\right)}{\max(\Delta,\delta_0)},
\end{aligned}$$

which implies (22) and also $\overline{R}_n \leq 24\sqrt{nK}$. Since Proposition 34 implies $\mathbb{E}R_n - \overline{R}_n \leq \sqrt{nK}$, we have proved (23). For (24), Proposition 36 implies

$$\mathbb{E}R_n - \overline{R}_n \leq \min\left(\frac{K}{\Delta}, \frac{\sqrt{nK}}{2}\right) \leq \frac{K\sqrt{75}}{2\max(\Delta,\delta_0)},$$

which implies

$$\mathbb{E}R_n \leq 23K\frac{\log\left(\frac{133n}{K}\max(\Delta,\delta_0)^2\right)}{\max(\Delta,\delta_0)}.$$

Appendix D. Pseudo-regret vs Expected Regret

The first two propositions hold for the four prediction games considered in this work and defined in Figure 1.

Proposition 33 *For deterministic adversaries, we have* $\mathbb{E}R_n = \overline{R}_n$. *For oblivious adversaries, we have*

$$\mathbb{E}R_n \leq \sup_{\textit{deterministic adversaries}} \overline{R}_n.$$

In particular, this means that the worst oblivious adversary for a forecaster cannot lead to a larger regret than the worst deterministic adversary.

Proof The first assertion is trivial. For the second one, let $\mathbb{E}_{\text{adv}}$ be the expectation with respect to the eventual randomization of the adversary and $\mathbb{E}_{\text{for}}$ be the expectation with respect to the randomization of the forecaster. For oblivious adversaries, we have $\mathbb{E}R_n = \mathbb{E}_{\text{adv}}\mathbb{E}_{\text{for}}R_n$, hence

$$\mathbb{E}R_n \leq \sup_{\text{deterministic adversaries}} \mathbb{E}_{\text{for}}R_n = \sup_{\text{deterministic adversaries}} \overline{R}_n.$$

■

While the previous proposition is useful for upper bounding the regret of a forecaster against the worst oblivious adversary, it does not say anything about the difference between the expected regret and the pseudo-regret for a given adversary. The next proposition gives an upper bound on this difference for fully oblivious adversaries, which are (oblivious) adversaries generating independently the reward vectors.

Proposition 34 *For fully oblivious adversaries, we have*

$$\mathbb{E}R_n - \overline{R}_n \leq \sqrt{\frac{n\log K}{2}},$$

and

$$\mathbb{E}R_n - \overline{R}_n \leq \sqrt{2\log(K)\max_i \mathbb{E}\sum_{t=1}^{n} g_{i,t}} + \frac{\log K}{3}.$$

Proof The proof is similar to the one of the upper bound on the expected supremum of a finite number of subgaussian random variables. We use the following lemma.

Lemma 35 *Let* $\lambda > 0$ *and* W *a random variable taking its values in* $[0,1]$. *We have*

$$\mathbb{E}\exp(\lambda W) \leq \exp\left[\big(\exp(\lambda) - 1\big)\mathbb{E}W\right].$$

Proof By convexity of the exponential function, we have $\exp(\lambda x) \leq 1 + \big(\exp(\lambda) - 1\big)x$ for any $x \in [0,1]$. So we have $\mathbb{E}\exp(\lambda W) \leq 1 + \big(\exp(\lambda) - 1\big)\mathbb{E}W$, hence Lemma 35. ■

Let $\lambda > 0$, then by Jensen's inequality and Lemma 35, we have

$$\begin{aligned}
\mathbb{E}\max_i \sum_{t=1}^n g_{i,t} &\leq \mathbb{E}\frac{1}{\lambda}\log\sum_{i=1}^K \exp\left(\lambda\sum_{t=1}^n g_{i,t}\right) \\
&\leq \frac{1}{\lambda}\log\sum_{i=1}^K \mathbb{E}\prod_{t=1}^n \exp(\lambda g_{i,t}) \\
&= \frac{1}{\lambda}\log\sum_{i=1}^K \prod_{t=1}^n \mathbb{E}\exp(\lambda g_{i,t}) \\
&\leq \frac{1}{\lambda}\log\sum_{i=1}^K \prod_{t=1}^n \exp\left([\exp(\lambda)-1]\mathbb{E}g_{i,t}\right) \\
&\leq \frac{\log K}{\lambda} + \frac{\exp(\lambda)-1}{\lambda}\max_i \mathbb{E}\sum_{t=1}^n g_{i,t}.
\end{aligned}$$

This implies

$$\mathbb{E}R_n - \overline{R}_n \leq \inf_{\lambda>0}\left(\frac{\log K}{\lambda} + \lambda\Theta(\lambda)\max_i \mathbb{E}\sum_{t=1}^n g_{i,t}\right),$$

where $\Theta(\lambda) = \frac{\exp(\lambda)-1-\lambda}{\lambda^2}$. By using Lemma 32, one obtains the second inequality of the theorem. Instead of using a variant of Bernstein's argument to control $\mathbb{E}\exp(\lambda g_{i,t})$, one can use Hoeffding's inequality. This leads to the first inequality by taking $\lambda = \sqrt{\frac{2\log K}{n}}$. ■

We can strengthen the previous result on the difference between the expected regret and the pseudo-regret when we consider the stochastic bandit game, in which the rewards coming from a given arm form an i.i.d. sequence. In particular, when there is a unique optimal arm, the following theorem states that the difference is exponentially small with n (instead of being of order $\sqrt{n}$).

Proposition 36 *For a given $\delta \geq 0$, let $I = \left\{i \in \{1,\ldots,K\} : \Delta_i \leq \delta\right\}$ be the set of arms "δ-close" to the optimal ones, and $J = \{1,\ldots,K\} \setminus I$ the remaining set of arms. In the stochastic bandit game, we have*

$$\mathbb{E}R_n - \overline{R}_n \leq \sqrt{\frac{n\log|I|}{2}} + \sum_{i\in J}\frac{1}{\Delta_i}\exp\left(-\frac{n\Delta_i^2}{2}\right),$$

and also

$$\begin{aligned}
\mathbb{E}R_n - \overline{R}_n \leq \sqrt{\frac{n\log|I|}{2}} + \sum_{i\in J}\Bigg\{ &\frac{8\sigma_i^2 + 4\Delta_i/3}{\Delta_i}\exp\left(-\frac{n\Delta_i^2}{8\sigma_i^2+4\Delta_i/3}\right) \\
&+ \frac{8\sigma_{i^*}^2 + 4\Delta_i/3}{\Delta_i}\exp\left(-\frac{n\Delta_i^2}{8\sigma_{i^*}^2+4\Delta_i/3}\right)\Bigg\},
\end{aligned}$$

where for any $j \in \{1,\ldots,K\}$, σ_j^2 denotes the variance of the reward distribution of arm j.

In particular when there exists a unique arm i^* such that $\Delta_{i^*} = 0$, we have

$$\mathbb{E}R_n - \overline{R}_n \leq \sum_{i\neq i^*}\frac{1}{\Delta_i}\exp\left(-\frac{n\Delta_i^2}{2}\right).$$

Note that the assumption on the uniqueness of the optimal arm in the last statement is necessary as we already discussed in Remark 25.

Proof Let $W_n^{(1)} = \max_{i\in I}\sum_{t=1}^n g_{i,t} - \sum_{t=1}^n g_{i^*,t}$ and $W_n^{(2)} = \max_{j\in\{1,\dots,K\}}\sum_{t=1}^n g_{j,t} - \max_{i\in I}\sum_{t=1}^n g_{i,t}$. We have $\mathbb{E}R_n - \overline{R}_n = \mathbb{E}W_n^{(1)} + \mathbb{E}W_n^{(2)}$. From the same argument as in the proof of Proposition 34, we have

$$\mathbb{E}W_n^{(1)} \le \sqrt{\frac{n\log|I|}{2}}.$$

Besides, we have

$$\begin{aligned}
\mathbb{E}W_n^{(2)} &= \int_0^{+\infty} \mathbb{P}(W_n^{(2)} > u)du \\
&\le \sum_{j\in J}\int_0^{+\infty} \mathbb{P}\left(\sum_{t=1}^n g_{j,t} - \max_{i\in I}\sum_{t=1}^n g_{i,t} > u\right)du \\
&\le \sum_{j\in J}\int_0^{+\infty} \mathbb{P}\left(G_{j,n} - G_{i^*,n} > u\right)du \\
&= \sum_{i\in J}\int_0^{+\infty} \mathbb{P}\left(G_{i,n} - \mathbb{E}G_{i,n} + \mathbb{E}G_{i^*,n} - G_{i^*,n} > u + n\Delta_i\right)du \\
&\le \sum_{i\in J}\int_0^{+\infty} \left\{\mathbb{P}\left(G_{i,n} - \mathbb{E}G_{i,n} > \frac{u+n\Delta_i}{2}\right) + \mathbb{P}\left(\mathbb{E}G_{i^*,n} - G_{i^*,n} > \frac{u+n\Delta_i}{2}\right)\right\}du.
\end{aligned}$$

This last integrand is upper bounded by $2\exp\left(-\frac{(u+n\Delta_i)^2}{2n}\right)$ from Hoeffding's inequality, and by $\exp\left(-\frac{(u+n\Delta_i)^2}{8n\sigma_i^2+4(u+n\Delta_i)/3}\right) + \exp\left(-\frac{(u+n\Delta_i)^2}{8n\sigma_{i^*}^2+4(u+n\Delta_i)/3}\right)$ from Bernstein's inequality. To control the two corresponding integrals, we note that for a nondecreasing convex function χ going to infinity at $+\infty$, we have

$$\int_x^{+\infty}\exp(-\chi(u))du \le \int_x^{+\infty}\frac{\chi'(u)}{\chi'(x)}\exp(-\chi(u))du = \frac{\exp(-\chi(x))}{\chi'(x)}.$$

We apply this inequality to the functions $r\mapsto\frac{r^2}{2n}$ and $r\mapsto\frac{r^2}{8n\sigma_i^2+4r/3}$ to obtain respectively

$$\mathbb{E}W_n^{(2)} \le 2\sum_{i\in J}\int_{n\Delta_i}^{+\infty}\exp\left(-\frac{u^2}{2n}\right)du \le \sum_{i\in J}\frac{1}{\Delta_i}\exp\left(-\frac{n\Delta_i^2}{2}\right),$$

and

$$\begin{aligned}
\int_{n\Delta_i}^{+\infty}\exp\left(-\frac{u^2}{8n\sigma_i^2+4u/3}\right)du &\le \frac{(8\sigma_i^2+4\Delta_i/3)^2}{\Delta_i(16\sigma_i^2+4\Delta_i/3)}\exp\left(-\frac{n\Delta_i^2}{8\sigma_i^2+4\Delta_i/3}\right) \\
&\le \frac{8\sigma_i^2+4\Delta_i/3}{\Delta_i}\exp\left(-\frac{n\Delta_i^2}{8\sigma_i^2+4\Delta_i/3}\right),
\end{aligned}$$

hence

$$\mathbb{E}W_n^{(2)} \le \sum_{i\in J}\left\{\frac{8\sigma_i^2+\frac{4\Delta_i}{3}}{\Delta_i}\exp\left(-\frac{n\Delta_i^2}{8\sigma_i^2+4\Delta_i/3}\right) + \frac{8\sigma_{i^*}^2+\frac{4\Delta_i}{3}}{\Delta_i}\exp\left(-\frac{n\Delta_i^2}{8\sigma_{i^*}^2+4\Delta_i/3}\right)\right\}.$$

■

References

C. Allenberg, P. Auer, L. Györfi, and G. Ottucsák. Hannan consistency in on-line learning in case of unbounded losses under partial monitoring. In *ALT*, volume 4264 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2006.

J.-Y. Audibert, R. Munos, and Cs. Szepesvári. Exploration-exploitation trade-off using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410:1876–1902, 2009.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, 1995.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning Journal*, 47(2-3):235–256, 2002a.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002b.

K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19:357–367, 1967.

N. Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. *Journal of Computer and System Sciences*, 59(3):392–411, 1999.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

N. Cesa-Bianchi, Y. Freund, D.P. Helmbold, D. Haussler, R.E. Schapire, and M.K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, 1997.

N. Cesa-Bianchi, G. Lugosi, and G. Stoltz. Minimizing regret with label efficient prediction. *IEEE: Transactions on Information Theory*, 51:2152–2162, 2005.

D. A. Freedman. On tail probabilities for martingales. *The Annals of Probability*, 3:100–118, 1975.

A. György and G. Ottucsák. Adaptive routing using expert advice. *Computer Journal-Oxford*, 49 (2):180–189, 2006.

D. Helmbold and S. Panizza. Some label efficient learning results. In *Proceedings of the 10th annual conference on Computational learning theory*, pages 218–230. ACM New York, NY, USA, 1997.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematics Society*, 58:527–535, 1952.

Journal of Machine Learning Research 11 (2010) 2837-2854 Submitted 6/09; Revised 3/10; Published 10/10

Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance

Nguyen Xuan Vinh* N.X.VINH@UNSW.EDU.AU
Julien Epps* J.EPPS@UNSW.EDU.AU
The University of New South Wales
Sydney, NSW 2052, Australia

James Bailey† JBAILEY@CSSE.UNIMELB.EDU.AU
The University of Melbourne
Vic. 3010, Australia

Editor: Marina Meilă

Abstract

Information theoretic measures form a fundamental class of measures for comparing clusterings, and have recently received increasing interest. Nevertheless, a number of questions concerning their properties and inter-relationships remain unresolved. In this paper, we perform an organized study of information theoretic measures for clustering comparison, including several existing popular measures in the literature, as well as some newly proposed ones. We discuss and prove their important properties, such as the metric property and the normalization property. We then highlight to the clustering community the importance of correcting information theoretic measures for chance, especially when the data size is small compared to the number of clusters present therein. Of the available information theoretic based measures, we advocate the normalized information distance (NID) as a general measure of choice, for it possesses concurrently several important properties, such as being both a metric and a normalized measure, admitting an exact analytical adjusted-for-chance form, and using the nominal $[0,1]$ range better than other normalized variants.

Keywords: clustering comparison, information theory, adjustment for chance, normalized information distance

1. Introduction

Clustering comparison measures play an important role in cluster analysis. Most often, such measures are used for external validation, that is, assessing the goodness of clustering solutions according to a "ground truth" clustering. Recent advances in cluster analysis have driven new algorithms, in which the clustering comparison measures are used actively in searching for good clustering solutions. One such example occurs in the context of ensemble (consensus) clustering, whose aim is to unify a set of clusterings, already obtained by some algorithms, into a single high quality one (Singh et al., 2009; Strehl and Ghosh, 2002; Charikar et al., 2003). A possible approach is to choose the clustering which shares the most information with all the other clusterings, such as in Strehl and Ghosh (2002). A measure is therefore needed to quantify the amount of information shared between clusterings, more specifically in this case, between the "centroid" clustering and all the

∗. Also in ATP Laboratory, National ICT Australia (NICTA).
†. Also in the Victoria Research Laboratory, National ICT Australia (NICTA).

other clusterings. Another example is in model selection by stability assessment (Ben-David et al., 2006; Shamir and Tishby, 2008). A possible realization of this scheme is to measure the average pairwise distances between all the clusterings obtained under some sort of perturbations (Vinh and Epps, 2009), hence requiring a clustering comparison measure.

Numerous measures for comparing clusterings have been proposed. Besides the class of *pair-counting based* and *set-matching based* measures, *information theoretic* measures form another fundamental class. In the clustering literature, such measures have been employed because of their strong mathematical foundation, and ability to detect non-linear similarities. For the particular purpose of clustering comparison, this class of measures has been popularized through the works of Strehl and Ghosh (2002) and Meilă (2005), and since then has been employed in various subsequent research (Fern and Brodley, 2003; He et al., 2008; Asur et al., 2007; Tumer and Agogino, 2008). In this context, the pioneering works of Meilă (2003, 2005, 2007) have shown a number of desirable theoretical properties of one of these measures—the *variation of information* (VI)—such as its metric property and its alignment with the lattice of partitions. Although having received considerable interest, in our opinion, the application of information theoretic measures for comparing clustering has been somewhat scattered. Apart from the VI which possesses a fairly comprehensive characterization, less is known about the *mutual information* and various forms of the so-called *normalized mutual information* (Strehl and Ghosh, 2002). The main technical contributions of this paper can be summarized as being three-fold:

1. We first review and make a coherent categorization of information theoretic similarity and distance measures for clustering comparison. We then discuss and prove their two important properties, namely the normalization and the metric properties. We show that among the prospective measures, the *normalized information distance* (NID) and the *normalized variation of information* (NVI) satisfy both these desirable properties.

2. We draw the attention of the clustering community towards the necessity of correcting information theoretic measures for chance in certain situations, derive analytical forms for the proposed adjusted-for-chance measures, and investigate their properties. Preliminary results regarding correcting information theoretic measures for chance have previously appeared in Vinh, Epps, and Bailey (2009). In this paper, we further analyze the large sample properties of the adjusted measures, and give a recommendation as to when adjustment is mostly needed.

3. Of the available information theoretic measures, we advocate the normalized information distance (NID) as a general purpose measure for comparing clusterings, which has the advantage of being both a metric and a normalized measure, admitting an exact analytical adjusted-for-chance form, and using better the nominal $[0,1]$ range. For ease of reading, we present the proofs of all results herein in the Appendix.

2. A Brief Review of Measures for Comparing Clusterings

Let S be a set of N data items, then a (partitional) clustering $\mathbf{U}$ on S is a way of partitioning S into non-overlap subsets $\{U_1, U_2, \ldots, U_R\}$, where $\cup_{i=1}^{R} U_i = S$ and $U_i \cap U_j = \emptyset$ for $i \neq j$. The information on the overlap between two clusterings $\mathbf{U} = \{U_1, U_2, \ldots, U_R\}$ and $\mathbf{V} = \{V_1, V_2, \ldots, V_C\}$ can be summarized in form of a $R \times C$ *contingency table* $M = [n_{ij}]_{j=1\ldots C}^{i=1\ldots R}$ as illustrated in Table 1, where n_{ij} denotes the number of objects that are common to clusters U_i and V_j.

Pair counting based measures are built upon counting pairs of items on which two clusterings agree or disagree. Specifically, the $\binom{N}{2}$ item pairs in S can be classified into one of the 4 types—N_{11}:

$\mathbf{U}\backslash\mathbf{V}$	V_1	V_2	$\dots$	V_C	Sums
U_1	n_{11}	n_{12}	$\dots$	n_{1C}	a_1
U_2	n_{21}	n_{22}	$\dots$	n_{2C}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
U_R	n_{R1}	n_{R2}	$\dots$	n_{RC}	a_R
Sums	b_1	b_2	$\dots$	b_C	$\sum_{ij} n_{ij} = N$

Table 1: The Contingency Table, $n_{ij} = |U_i \cap V_j|$

the number of pairs that are in the same cluster in both $\mathbf{U}$ and $\mathbf{V}$; N_{00}: the number of pairs that are in different clusters in both $\mathbf{U}$ and $\mathbf{V}$; N_{01}: the number of pairs that are in the same cluster in $\mathbf{U}$ but in different clusters in $\mathbf{V}$; and N_{10}: the number of pairs that are in different clusters in $\mathbf{U}$ but in the same cluster in $\mathbf{V}$—that can be calculated using the n_{ij}'s (Hubert and Arabie, 1985). Intuitively, N_{11} and N_{00} can be used as indicators of agreement between $\mathbf{U}$ and $\mathbf{V}$, while N_{01} and N_{10} can be used as disagreement indicators. A well known index of this class is the Rand index (RI, Rand 1971), defined straightforwardly as $\mathrm{RI}(\mathbf{U},\mathbf{V}) = (N_{00}+N_{11})/\binom{N}{2}$, which lies in the nominal range of [0,1]. In practice however, the RI often lies within the narrower range of [0.5,1]. Also, its baseline value can be high and does not take on a constant value. For these reasons, the RI has been mostly used in it adjusted form, known as the adjusted Rand index (ARI, Hubert and Arabie 1985):

$$\mathrm{ARI}(\mathbf{U},\mathbf{V}) = \frac{2(N_{00}N_{11} - N_{01}N_{10})}{(N_{00}+N_{01})(N_{01}+N_{11}) + (N_{00}+N_{10})(N_{10}+N_{11})}.$$

The ARI is bounded above by 1, and equals 0 when the RI equals its expected value (under the generalized hypergeometric distribution assumption for randomness). Besides the ARI, there are many other, possibly less popular, measures in this class. Albatineh et al. (2006) made a comprehensive list of 22 different indices of this type, a number which is large enough to make the task of choosing an appropriate measure difficult and confusing. Their work, and subsequent extension of Warrens (2008), showed that after correction for chance, some of these measures become equivalent. Despite the existence of numerous measures, the ARI remains the most well-known and widely used (Steinley, 2004). Therefore, in this work, we take it as the representative of this class for comparison with other measures. Although the ARI has been mainly used in its similarity form, it can be easily shown that its distance version, that is, $1-\mathrm{ARI}$, is not a proper metric.

Set matching based measures, as their name suggests, are based on finding matches between clusters in the two clusterings. A popular measure is the classification error rate which is often employed in supervised learning. Several other indices are discussed in Meilă (2007), all suffering from two major problems which have long been known in the clustering comparing literature (Dom, 2001; Steinley, 2004; Meilă, 2007) namely: (i) the number of clusters in the two clusterings may be different, making this approach problematic, since there are some clusters which are put outside consideration; and (ii) even when the numbers of clusters are the same, the unmatched part of each matched cluster pair is still put outside consideration. Due to the problems with this class of indices, we shall not consider them further in this paper.

Information theoretic based measures are built upon fundamental concepts from information theory (Cover and Thomas, 1991). Given two clusterings $\mathbf{U}$ and $\mathbf{V}$, their entropies, joint entropy,

conditional entropies and mutual information (MI) are defined naturally via the marginal and joint distributions of data items in $\mathbf{U}$ and $\mathbf{V}$ respectively as:

$$\begin{aligned} H(\mathbf{U}) &= -\sum_{i=1}^{R} \frac{a_i}{N} \log \frac{a_i}{N}, \\ H(\mathbf{U},\mathbf{V}) &= -\sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}}{N}, \\ H(\mathbf{U}|\mathbf{V}) &= -\sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}/N}{b_j/N}, \\ I(\mathbf{U},\mathbf{V}) &= \sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}/N}{a_i b_j/N^2}. \end{aligned}$$

The MI measures the information that $\mathbf{U}$ and $\mathbf{V}$ share: it tells us how much knowing one of these clusterings reduces our uncertainty about the other. From a communication theory point of view, the above-defined quantities can be interpreted as follows. Suppose we need to transmit all the cluster labels in $\mathbf{U}$ on a communication channel, then $H(\mathbf{U})$ can be interpreted as the average amount of information, for example, in bits, needed to encode the cluster label of each data point according to $\mathbf{U}$. Now suppose that $\mathbf{V}$ is made available to the receiver, then $H(\mathbf{U}|\mathbf{V})$ denotes the average number of bits needed to transmit each label in $\mathbf{U}$ if $\mathbf{V}$ is already known. We are interested in how seeing how much $H(\mathbf{U}|\mathbf{V})$ is smaller than $H(\mathbf{U})$, that is, how much the knowledge of $\mathbf{V}$ helps us to reduce the number of bits needed to encode $\mathbf{U}$. This can be quantified in terms of the mutual information $H(\mathbf{U}) - H(\mathbf{U}|\mathbf{V}) = I(\mathbf{U},\mathbf{V})$. The knowledge of $\mathbf{V}$ thus helps us to reduce the number of bits needed to encode each cluster label in $\mathbf{U}$ by an amount of $I(\mathbf{U},\mathbf{V})$ bits. In the reverse direction we also have $I(\mathbf{U},\mathbf{V}) = H(\mathbf{V}) - H(\mathbf{V}|\mathbf{U})$. Clearly, the higher the MI, the more useful the information in $\mathbf{V}$ helps us to predict the cluster labels in $\mathbf{U}$ and vice-versa.

Before closing this section, we list several generally desirable properties of a clustering comparison measure. This list is not meant to be exhaustive, and particular applications might require other specific properties.

- *Metric property*: the metric property requires that a distance measure satisfy the properties of a true metric, namely positive definiteness, symmetry and triangle inequality. As the most basic benefit, the metric property conforms to our intuition of distance (Meilă, 2007). Furthermore, it is important if one would like to study, either the structure of, or design algorithms for the complex space of clusterings, as many nice theoretical results already exist for metric spaces.

- *Normalization*: the normalization property requires that the range of a similarity or distance measure lies within a fixed range, for example, [-1,1] or [0,1]. Normalization facilitates interpretation and comparison across different conditions (Strehl and Ghosh, 2002; Luo et al., 2009), where unbounded measures might have different ranges. Also, normalization has been shown to improve the sensitiveness of certain measures, such as the MI, with respect to the difference in cluster distribution in the two clusterings (Wu et al., 2009). The fact that all of the 22 different pair counting based measures discussed in Albatineh et al. (2006) are normalized, further stresses the particular interest of the clustering community in this property.

- *Constant baseline property*: for a similarity measure, its expected value between pairs of independent clusterings, for example, clusterings sampled independently at random, should be a constant. Ideally this baseline value should be zero, indicating no similarity. The Rand index is an example of a similarity index which does not satisfy this rather intuitive property, the reason why it has been mainly used in its adjusted form.

3. Information Theoretic Based Measures - Variants and Properties

Name	Expression	Range	Related sources
Mutual Information (MI)	$I(\mathbf{U},\mathbf{V})$	$[0,\min\{H(\mathbf{U}),H(\mathbf{V})\}]$	Banerjee et al. (2005)
Normalized MI (NMI)			
NMI_{joint}	$\frac{I(\mathbf{U},\mathbf{V})}{H(\mathbf{U},\mathbf{V})}$	[0,1]	Yao (2003)
NMI_{max}	$\frac{I(\mathbf{U},\mathbf{V})}{\max\{H(\mathbf{U}),H(\mathbf{V})\}}$	[0,1]	Kvalseth (1987)
NMI_{sum}	$\frac{2I(\mathbf{U},\mathbf{V})}{H(\mathbf{U})+H(\mathbf{V})}$	[0,1]	Kvalseth (1987)
NMI_{sqrt}	$\frac{I(\mathbf{U},\mathbf{V})}{\sqrt{H(\mathbf{U})H(\mathbf{V})}}$	[0,1]	Strehl and Ghosh (2002)
NMI_{min}	$\frac{I(\mathbf{U},\mathbf{V})}{\min\{H(\mathbf{U}),H(\mathbf{V})\}}$	[0,1]	Kvalseth (1987) Liu et al. (2008)
Adjusted-for-Chance MI (see Section 4)			
$\mathrm{AMI}_{max}{}^{\dagger}$	$\frac{I(\mathbf{U},\mathbf{V})-E\{I(\mathbf{U},\mathbf{V})\}}{\max\{H(\mathbf{U}),H(\mathbf{V})\}-E\{I(\mathbf{U},\mathbf{V})\}}$	$[0,1]^{*}$	
$\mathrm{AMI}_{sum}{}^{\dagger}$	$\frac{I(\mathbf{U},\mathbf{V})-E\{I(\mathbf{U},\mathbf{V})\}}{\frac{1}{2}[H(\mathbf{U})+H(\mathbf{V})]-E\{I(\mathbf{U},\mathbf{V})\}}$	$[0,1]^{*}$	
$\mathrm{AMI}_{sqrt}{}^{\dagger}$	$\frac{I(\mathbf{U},\mathbf{V})-E\{I(\mathbf{U},\mathbf{V})\}}{\sqrt{H(\mathbf{U})H(\mathbf{V})}-E\{I(\mathbf{U},\mathbf{V})\}}$	$[0,1]^{*}$	
$\mathrm{AMI}_{min}{}^{\dagger}$	$\frac{I(\mathbf{U},\mathbf{V})-E\{I(\mathbf{U},\mathbf{V})\}}{\min\{H(\mathbf{U}),H(\mathbf{V})\}-E\{I(\mathbf{U},\mathbf{V})\}}$	$[0,1]^{*}$	

*These measures are normalized in a stochastic sense, being equal to 1 if the (unadjusted) measures equal their value as expected by chance agreement. †Our proposed measures.

Table 2: Information theoretic-based similarity measures

Similarity measures: the mutual information (MI), a non-negative quantity, can be employed as the most basic similarity measure. Based on the observation that the MI is upper-bounded by the following quantities:

$$I(\mathbf{U},\mathbf{V}) \leq \min\{H(\mathbf{U}),H(\mathbf{V})\} \leq \sqrt{H(\mathbf{U})H(\mathbf{V})} \leq \frac{1}{2}(H(\mathbf{U})+H(\mathbf{V})) \leq \max\{H(\mathbf{U}),H(\mathbf{V})\} \leq H(\mathbf{U},\mathbf{V}), \quad (1)$$

we can derive several normalized versions of the mutual information (NMI) as listed in Table 2. All the normalized variants are bounded in [0,1], equaling 1 when the two clusterings are identical, and 0 when they are independent, that is, sharing no information about each other. In the latter case, the contingency table takes the form of the so-called "independence table" where $n_{ij} = |U_i||V_j|/N$ for all i, j. The MI and some of its normalized versions have been used in the clustering literature as similarity measures between objects in general (see, for example, Yao, 2003 and references therein). For the particular purpose of clustering comparison, Banerjee et al. (2005) employed the unnormalized MI. Strehl and Ghosh (2002) on the other hand made use of the NMI_{sqrt} normalized version, which has also been used in several follow-up works in the context of ensemble clustering (Fern and Brodley, 2003; He et al., 2008; Asur et al., 2007; Tumer and Agogino, 2008).

Name	Expression	Range	Metric	Related sources
Unnormalized distance measures				
D_{joint} (Variation of Information)	$H(\mathbf{U},\mathbf{V}) - I(\mathbf{U},\mathbf{V})$	[0,log$N$]	✓	Yao (2003) Meilă (2005)
D_{max}	$\max\{H(\mathbf{U}),H(\mathbf{V})\} - I(\mathbf{U},\mathbf{V})$	[0,log$N$]	✓	
$D_{sum}(\equiv 1/2 D_{joint})$	$\frac{1}{2}[H(\mathbf{U})+H(\mathbf{V})] - I(\mathbf{U},\mathbf{V})$	[0,log$N$]	✓	
D_{sqrt}	$\sqrt{H(\mathbf{U})H(\mathbf{V})} - I(\mathbf{U},\mathbf{V})$	[0,log$N$]	✗	
D_{min}	$\min\{H(\mathbf{U}),H(\mathbf{V})\} - I(\mathbf{U},\mathbf{V})$	[0,log$N$]	✗	
Normalized distance measures				
d_{joint} (Normalized VI)	$1-\frac{I(\mathbf{U},\mathbf{V})}{H(\mathbf{U},\mathbf{V})}$	[0,1]	✓	Kraskov et al. (2005)
d_{max} (Normalized Information Distance)	$1-\frac{I(\mathbf{U},\mathbf{V})}{\max\{H(\mathbf{U}),H(\mathbf{V})\}}$	[0,1]	✓	Kraskov et al. (2005)
d_{sum}	$1-\frac{2I(\mathbf{U},\mathbf{V})}{H(\mathbf{U})+H(\mathbf{V})}$	[0,1]	✗	
d_{sqrt}	$1-\frac{I(\mathbf{U},\mathbf{V})}{\sqrt{\{H(\mathbf{U}),H(\mathbf{V})\}}}$	[0,1]	✗	
d_{min}	$1-\frac{I(\mathbf{U},\mathbf{V})}{\min\{H(\mathbf{U}),H(\mathbf{V})\}}$	[0,1]	✗	
Adjusted-for-Chance distance measures (see Section 4)				
$Ad_{max}{}^{\dagger}$	$1-\mathrm{AMI}_{max}$	$[0,1]^*$	✗	
$Ad_{sum}{}^{\dagger}$	$1-\mathrm{AMI}_{sum}$	$[0,1]^*$	✗	
$Ad_{sqrt}{}^{\dagger}$	$1-\mathrm{AMI}_{sqrt}$	$[0,1]^*$	✗	
$Ad_{min}{}^{\dagger}$	$1-\mathrm{AMI}_{min}$	$[0,1]^*$	✗	

*These measures are normalized in a stochastic sense, being equal to 0 if the (unadjusted) measures equal their value as expected by chance agreement. †Our proposed measures. D denotes an unnormalized distance measure, d denotes a normalized distance measure

Table 3: Information theoretic-based distance measures

Distance measures: based on the five upper bounds for $I(\mathbf{U},\mathbf{V})$ given in (1), we can define five distance measures, namely $D_{joint}, D_{max}, D_{sum}, D_{sqrt}$ and D_{min}, as detailed in Table 3. However, it can be seen that $D_{joint} = 2D_{sum}$,[1] and these two measures have been known in the clustering literature as the variation of information—VI (Meilă, 2005). The fact that D_{joint} (and hence D_{sum}) is a true metric is a well known result (Meilă, 2005). In addition, we also present the following new results (see Appendix for proof):

Theorem 1 *D_{max} is a metric.*

Theorem 2 *D_{min} and D_{sqrt} are **not** metrics.*

The negative result given in Theorem 2 is indeed helpful in narrowing our search scope for a reasonable distance measure. So far, D_{max} and D_{joint} (D_{sum}) are potential candidates. These distance measures do not have a fixed upper bound however, and we are therefore seeking some normalized variants. By dividing each distance measure by its corresponding upper bound we can define five normalized variants as detailed in Table 3, which are actually the unit-complements of the corresponding NMI variants, for example, $d_{joint} = 1 - \mathrm{NMI}_{joint}$. We now state the following properties of the normalized distance measures:

Theorem 3 *The normalized variation of information, d_{joint}, is a metric*

Theorem 4 *The normalized information distance, d_{max}, is a metric*

1. $2D_{sum}(\mathbf{U},\mathbf{V}) = H(\mathbf{U})+H(\mathbf{V})-2I(\mathbf{U},\mathbf{V}) = [H(\mathbf{U})+H(\mathbf{V})-I(\mathbf{U},\mathbf{V})] - I(\mathbf{U},\mathbf{V}) = H(\mathbf{U},\mathbf{V}) - I(\mathbf{U},\mathbf{V}) = D_{joint}(\mathbf{U},\mathbf{V})$.

Theorem 5 *The normalized distance measures d_{min}, d_{sum} and d_{sqrt}, are* ***not*** *metrics.*

The proofs for Theorem 3 and 4 was presented in an unofficially extended version of Kraskov et al. (2005).[2] Unfortunately, their proof for Theorem 4 was erroneous.[3] Since these are two interesting results, we give our shortened proof for Theorem 3 and a corrected proof for Theorem 4 in the Appendix. The negative results in Theorem 5 are again useful in narrowing our scope looking for a good candidate. From our discussion so far, we can now identify two promising candidates: d_{joint} and d_{max}. Since the variation of information—D_{joint}—is the unnormalized version of d_{joint}, we shall name d_{joint} the normalized variation of information (NVI). d_{max} has not been named in the literature, therefore we name it after its well known analogue in Kolmogorov complexity theory (Li et al., 2004), the normalized information distance (NID). Both the NVI and NID have the remarkable property of being both a metric and a normalized measure. We note that Meilă (2007) proposed normalized variants for the VI, such as: $V(\mathbf{U},\mathbf{V}) = \frac{1}{\log N}\mathrm{VI}(\mathbf{U},\mathbf{V})$ or: $V_{K^*}(\mathbf{U},\mathbf{V}) = \frac{1}{2\log K^*}\mathrm{VI}(\mathbf{U},\mathbf{V})$ when the number of clusters in both $\mathbf{U}$ and $\mathbf{V}$ is bounded by the same constant $K^* < \sqrt{N}$. The bounds of $\log N$ and $2\log K^*$ are not as strict as $H(\mathbf{U},\mathbf{V})$ however,[4] thus the useful range of these normalized VI variants is narrower than that of d_{joint}. The joint entropy $H(\mathbf{U},\mathbf{V})$ provides a stricter upper bound, enabling d_{joint} to better exploit the [0,1] range, while still retaining the metric property. It is noted that since $\max\{H(\mathbf{U}),H(\mathbf{V})\}$ is yet a tighter upper bound for $\mathrm{MI}(\mathbf{U},\mathbf{V})$ than $H(\mathbf{U},\mathbf{V})$, d_{max} is generally more preferable to d_{joint} since it can even better use the nominal range of $[0,1]$. A subtle point regarding normalization by quantities such as $\max\{H(\mathbf{U}),H(\mathbf{V})\}$ and $H(\mathbf{U},\mathbf{V})$, as has been brought to our attention by the Editor, is their potential side effects on the normalization process. For validation purpose for example, if $\mathbf{U}$ is the ground-truth, and $\mathbf{V}$ is the clustering obtained by some algorithm, then the normalization also depends on $\mathbf{V}$. Thus, while random quantities such as $\max\{H(\mathbf{U}),H(\mathbf{V})\}$ and $H(\mathbf{U},\mathbf{V})$ provide tighter bounds, their effect on the normalization process is not as clear as looser, fixed bounds such as $\log N$ and $2\log K^*$.

4. Adjustment for Chance

In this section we inspect the proposed information theoretic measures with respect to the third desirable property, that is, the *constant baseline property*. We shall first point out that, just like the well-known Rand index, the baseline value of information theoretic measures does not take on a constant value, and thus adjustment for chance will be needed in certain situations. Let us consider the following two motivating examples:

1) Example 1 - Distance to a "true" clustering: given a ground-truth clustering $\mathbf{U}$ with K_{true} clusters, we need to assess the goodness of two clusterings $\mathbf{V}$ with C clusters, and $\mathbf{V}'$ with C' clusters. If $C = C'$ then the situation would be quite simple. Since the setting is the same for both $\mathbf{V}$ and $\mathbf{V}'$, we expect the comparison to be "fair" under any particular measure. However if $C \neq C'$, the situation becomes more complicated. We set up an experiment as follows: consider a set of N data points, let the number of clusters K vary from 2 to K_{max} and suppose that the true clustering has $K_{true} = [K_{max}/2]$ clusters. Now for each value of K, generate 10,000 random clusterings and calculate the average MI, NMI_{max}, VI, RI and ARI between those clusterings to a fixed, random

2. Available online at `http://arxiv.org/abs/q-bio/0311039v2`.
3. In their case 1, $D'(Z,Y)$ is in fact not equal to $H(Z|Y)/H(Y)$.
4. If $N \leq RC$ then $H(\mathbf{U},\mathbf{V}) \leq \log N$, with the equality attained only when cells of the contingency table contain only either 1 or 0. If $N > RC$ then $H(\mathbf{U},\mathbf{V}) \leq \log(RC) \leq \log(K^*K^*) = 2\log K^*$.

clustering chosen as the "true" clustering. The results for two combinations of (N, K_{true}) are given in Fig. 1(a,b). It can be observed that the unadjusted measures such as the RI, MI and NMI (VI) monotonically increase (decreases) as K increases. Thus even by selecting totally at random, a 7-cluster solution would have a greater chance to outperform a 3-cluster solution, although there isn't any difference in the clustering generation methodology. A corrected-for-chance measure, such as the ARI, on the other hand, has a baseline value always close to zero, and appears not to be biased in favor of any particular value of K. The same issue is observed with all other variants of the NMI (data not shown). Thus for this example, an adjusted-for-chance version of the MI is desirable.

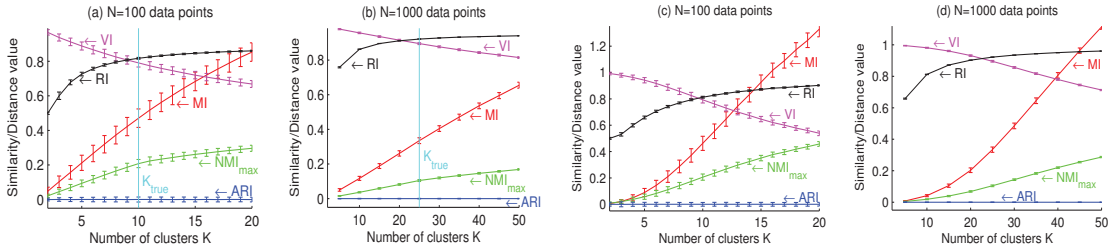


Figure 1: (a,b) Average distance between sets of random clusterings to a "true" clustering (c,d) Average pairwise distance in a set of random clusterings. Error bars denote standard deviation.

2) Example 2 - Determining the number of clusters via consensus (ensemble) clustering: in an era where a huge number of clustering algorithms exist, the consensus clustering idea (Monti et al., 2003; Strehl and Ghosh, 2002; Yu et al., 2007) has recently received increasing interest. Consensus clustering is not just another clustering algorithm: it rather provides a framework for unifying the knowledge obtained from other algorithms. Given a data set, consensus clustering employs one or several clustering algorithms to generate a set of clustering solutions on either the original data set or its perturbed versions. From these clustering solutions, consensus clustering aims to choose a robust and high quality representative clustering. Although the main objective of consensus clustering is to discover a high quality cluster structure, closer inspection of the set of clusterings obtained can often give valuable information about the appropriate number of clusters present. More specifically, we have empirically observed the following: in regard to the set of clusterings obtained, when the specified number of clusters coincides with the true number of clusters, this set has a tendency to be less diverse. This is an indication of the robustness of the obtained cluster structure. To quantify this diversity we have recently developed a novel index (Vinh and Epps, 2009), namely the *consensus index* (CI), which is built upon a suitable clustering similarity measure. Given a value of K, suppose we have generated a set of B clustering solutions $\mathcal{U}_K = \{\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_B\}$, each with K clusters. We define the consensus index of $\mathcal{U}_K$ as:

$$\mathrm{CI}(\mathcal{U}_K) = \frac{\sum_{i<j} \mathrm{AM}(\mathbf{U}_i, \mathbf{U}_j)}{B(B-1)/2}$$

where the agreement measure AM is a suitable clustering similarity index. Thus, the CI quantifies the average pairwise agreement in $\mathcal{U}_K$. The optimal number of clusters K^* is chosen as which that maximizes CI, that is, $K^* = \arg\max_{K=2\ldots K_{max}} \mathrm{CI}(\mathcal{U}_K)$. In this setting, a normalized measure is preferable for its equalized range at different values of K. We performed an experiment as follows:

given N data points, randomly assign each data point into one of the K clusters with equal probability and check to ensure that the final clustering contains exactly K clusters. For each K, repeat this 200 times to create 200 random clusterings, then calculate the average values of the MI, VI, NMI_{max}, RI and ARI between all 19,900 clustering pairs. Typical experimental results are presented in Fig. 1(c,d). It can be observed that for a given data set, the average MI, NMI and RI (VI) values between random clusterings tend to increase (decrease) as the number of clusters increases, while the average value of the ARI is always close to zero. When the ratio of N/K is large, the average value for NMI is reasonably close to zero, but grows as N/K becomes smaller. This is clearly an unwanted effect, since a consensus index built upon the MI, NMI and VI would be biased in favour of a larger number of clusters. Thus in this situation, an adjusted-for-chance version of the MI is again important.

4.1 The Proposed Adjusted Measures

To correct the measures for randomness it is necessary to specify a model according to which random partitions are generated. Such a common model is the "permutation model" (Lancaster, 1969, p. 214), in which clusterings are generated randomly subject to having a fixed number of clusters and points in each clusters. Using this model, which was also adopted by Hubert and Arabie (1985) for the ARI, we have previously shown (Vinh et al., 2009) that the expected mutual information between two clusterings $\mathbf{U}$ and $\mathbf{V}$ is:

$$\mathbf{E}\{I(\mathbf{U},\mathbf{V})\} = \sum_{i=1}^{R}\sum_{j=1}^{C}\sum_{n_{ij}=\max(a_i+b_j-N,0)}^{\min(a_i,b_j)} \frac{n_{ij}}{N}\log\left(\frac{N.n_{ij}}{a_i b_j}\right)\frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(N-a_i-b_j+n_{ij})!}. \quad (2)$$

As suggested by Hubert and Arabie (1985), the general form of a similarity index corrected for chance is given by:

$$\text{Adjusted\_Index} = \frac{\text{Index} - \text{Expected\_Index}}{\text{Max\_Index} - \text{Expected\_Index}}, \quad (3)$$

which is upper-bounded by 1 and equals 0 when the index equals its expected value. Having calculated the expectation of the MI, we propose the adjusted form, which we call the *adjusted mutual information* (AMI), for the normalized mutual information according to (3). For example, taking the NMI_{max} we have:

$$\text{AMI}_{max}(\mathbf{U},\mathbf{V}) = \frac{\text{NMI}_{max}(\mathbf{U},\mathbf{V}) - \mathbf{E}\{\text{NMI}_{max}(\mathbf{U},\mathbf{V})\}}{1 - \mathbf{E}\{\text{NMI}_{max}(\mathbf{U},\mathbf{V})\}} = \frac{I(\mathbf{U},\mathbf{V}) - \mathbf{E}\{I(\mathbf{U},\mathbf{V})\}}{\max\{H(\mathbf{U}),H(\mathbf{V})\} - \mathbf{E}\{I(\mathbf{U},\mathbf{V})\}}.$$

Similarly, other adjusted *similarity* measures are listed in Table 2. It can be seen that the adjusted-for-chance forms of the MI are all normalized in a stochastic sense. Specifically, the AMI equals 1 when the two clusterings are identical, and 0 when the MI between the two clusterings equals its expected value. The adjusted forms for the *distance* measures, listed in Table 2, are again the unit-complements of the corresponding adjusted *similarity* measures, for example, $Ad_{max} = 1 - \text{AMI}_{max}$, and are also normalized in a stochastic sense. Following the naming scheme that we have adopted throughout in this paper, we name Ad_{max} the adjusted information distance. It is noted that at this stage, we have not been able to derive an analytical solution for the adjusted form for the normalized variation of information (d_{joint}) measure. The derivation of the expected value for this measure appears to be more involved observing that $I(\mathbf{U},\mathbf{V})$ is present in both the numerator and denominator ($H(\mathbf{U},\mathbf{V}) = H(\mathbf{U}) + H(\mathbf{V}) - I(\mathbf{U},\mathbf{V})$). We repeat the experiments described in examples 1 and 2, this time with the adjusted version of the NMI_{max}. Now it can be seen from Fig. 2 that just like the ARI, the AMI_{max} baseline values are close to zero. It is noted that in these experiments, we did not

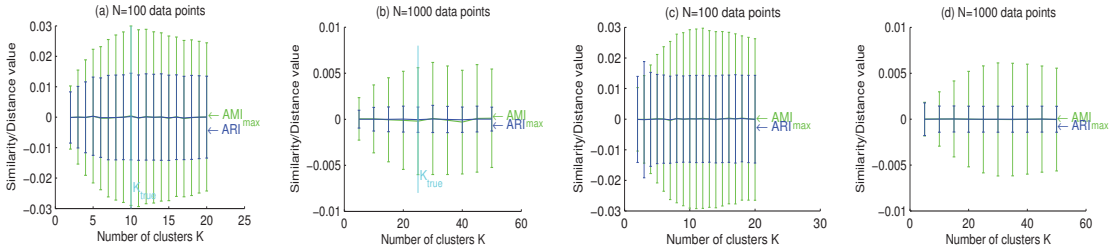


Figure 2: (a,b) Average distance between sets of random clusterings to a "true" clustering (c,d) Average pairwise distance in a set of random clusterings. Error bars denote standard deviation.

require the marginals of the contingency table to be fixed as per the assumption of the generalized hypergeometric model of randomness. Nevertheless, the adjusted measures still exhibit the desired behavior.

4.2 Properties of the Adjusted Measures

While admitting a constant baseline, the proposed adjusted-for-chance measures are, unfortunately, not proper metrics:

Theorem 6 *The adjusted measures $Ad_{max}, Ad_{sum}, Ad_{sqrt}$ and Ad_{min} are **not** metrics.*

There is thus a trade-off between the metric property and correction for chance, and the user should decide which property is of higher priority. Fortunately, during our experiments with the AMI, we have observed that when the data contain a fairly large number of items as compared to the number of clusters, for example, $N/K \geq 100$, then the expected mutual information is fairly close to zero, as can be seen in Fig. 1, suggesting scenarios where adjustment for chance is not of utmost necessity. The following results formalize this observation:

Theorem 7 *Some upper bounds for the expected mutual information between two random clusterings $\boldsymbol{U}$ and $\boldsymbol{V}$ (on a data set of N data items, with R and C clusters respectively), under the hypergeometric distribution model of randomness are given by the followings:*

$$\boldsymbol{E}\{I(\boldsymbol{U},\boldsymbol{V})\} \leq \sum_{i=1}^{R}\sum_{j=1}^{C}\frac{a_i b_j}{N^2}\log\left(\frac{N(a_i-1)(b_j-1)}{(N-1)a_i b_j}+\frac{N}{a_i b_j}\right) \leq \log\left(\frac{N+RC-R-C}{N-1}\right). \quad (4)$$

These bounds shed light on the large sample property of the adjusted measures. The following result trivially follows:

Corollary 1 *Given R and C fixed, $\lim_{N\to\infty}\boldsymbol{E}\{I(\boldsymbol{U},\boldsymbol{V})\} = 0$, and thus the adjusted measures tend toward the normalized measures.*

Also, these bounds give useful information on whether adjustment for chance is needed. For example, on a data set of 100 data items and two clusterings $\mathbf{U}$ and $\mathbf{V}$, each having 10 clusters with sizes of $[10,10,10,10,10,10,10,10,10,10]$ and $[2,4,6,8,10,10,12,14,16,18]$ respectively, the expected MI and its upper bounds according to (2) and (4) are $\mathbf{E}\{I(\mathbf{U},\mathbf{V})\} = 0.4618 < 0.5584 <$

0.5978. As the maximum MI value is only $\log(10) = 2.3$, correction for chance is needed since the baseline is high. However, if the data size increases ten-fold to 1000 items, keeping the same number of clusters and cluster distribution, the two upper bounds are 0.0764 and 0.0780 respectively, which can be considered small enough for many applications, therefore adjustment for chance might be omitted.

4.3 An Example Application

As per our analysis, adjustment for chance for information theoretic measures is mostly needed when the number of data items is relatively small compared to the number of clusters. One such prominent example is in microarray data analysis, where biological samples are clustered using gene expression data. Due to the high cost of preparing and collecting microarray data, each class, for example, of tumor, might contain only as few as several samples. In this section we demonstrate the use of the consensus index to estimate the number of clusters in microarray data. Eight synthetic and real microarray data sets are drawn from Monti et al. (2003), as detailed in Table 4 (see the original publication for preprocessing issues). A quick check upon the (higher) upper bound of the expected MI on these data sets suggests that correction for chance will be needed, for example, on the Leukemia data set, as $K(= R = C)$ grows from 2 to 10, this upper bound grows from 0.03 to 1.16.

Simulated Data	#Classes	#Samples	#Genes	Real data	#Classes	#Samples	#Genes
Gaussian3	3	60	600	Leukemia	3	38	999
Gaussian5	5	500	2	Novartis	4	103	1000
Simulated4	4	40	600	Lung cancer	4+	197	1000
Simulated6	6-7	60	600	Normal tissues	13	90	1277

Table 4: Microarray data sets summary, source: Monti et al. (2003)

In Vinh and Epps (2009) we have shown that the CI, coupled with sub-sampling as the perturbation method, gives useful information on the appropriate number of clusters in microarray data. Herein, we experimented with random projection as the perturbation scheme. More specifically, the original data set was projected on a random set of 80% of the genes, and the K-means clustering algorithm was run with random initialization on the projected data set. For each value of K, 100 of such clustering solutions were created and the CI's for 6 measures, namely RI, ARI, MI, VI, NMI_{max} and AMI_{max} were calculated. Ideally we expect to see a strong global peak at the true number of cluster K_{true}. From Fig. 3(a) it can be observed that the unadjusted MI has a strong bias with respect to the number of clusters, increasing monotonically as K increases. Similar behavior was observed with all other data sets and therefore MI is not an appropriate measure for this purpose. For ease of presentation, we have excluded the MI from Fig. 3(b-h). The effect of adjustment for chance can be clearly observed in Fig. 3(c,d,e,h). Agreement by chance inflates the CI score of the unadjusted measures (RI, NMI, VI) in such cases, and can lead to incorrect estimation. The CI of the adjusted measures (ARI, AMI) correctly estimates the number of clusters in all synthetic data sets with high confidence, whereas on real data sets it gives correct estimations on the Leukemia and Normal tissues data set. The CI suggests only 3 clusters on the Novartis while the assumed number of clusters is 4. The Lung cancer data set is an example where human experts are not yet confident on the true number of clusters present (4+ clusters), while the CI gives a local peak at 6 clusters. These results are concordant with our previous finding (Vinh and Epps, 2009). The fact that the CI

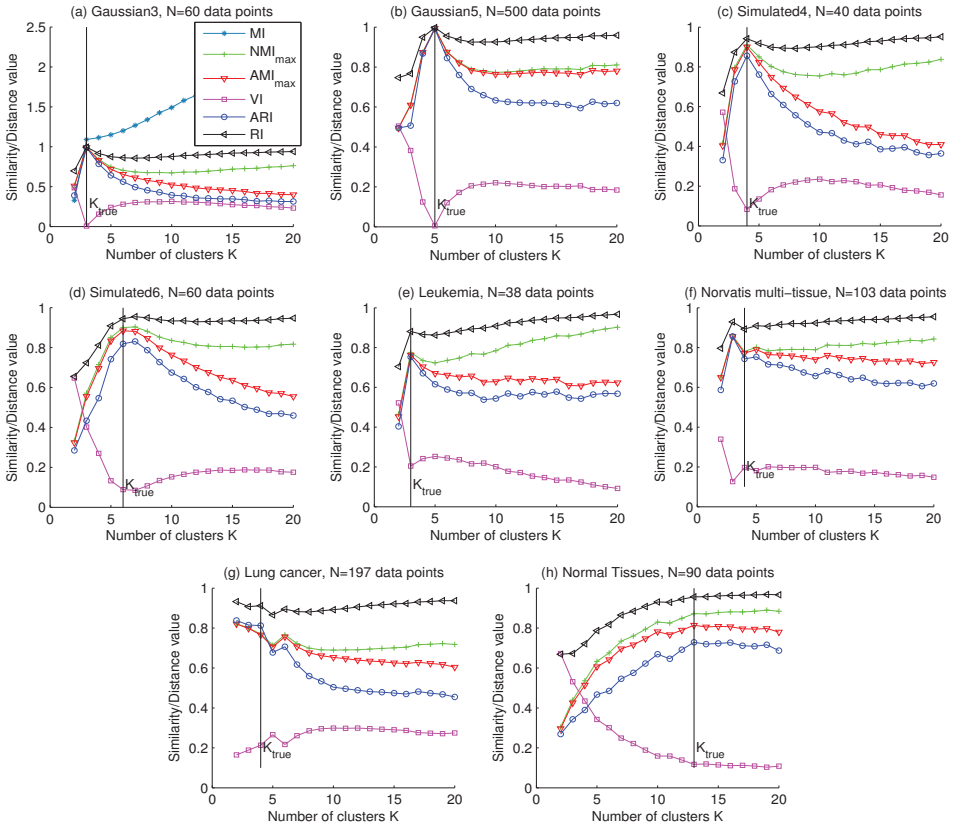


Figure 3: Consensus Index on microarray data sets.

showing global or local peaks at or near the presumably true number of clusters as attributed by the respective authors calls for further investigation on both the biological side (re-verifying the number of clusters), and the CI side (the behaviour of the index with respect to the structure of the data set, for example, the data set might contain a hierarchy of clusters and thus the CI may exhibit local peaks corresponding to such structures).

5. Related Work

Meilă (2005) considered clustering comparison measures with respect to their alignment with the lattice of partitions. In addition to the metric property, she considered three more properties namely *additivity with respect to refinement*, *additivity with respect to the join* and *convex additivity*, and showed that the VI satisfies all these properties. Unfortunately, none of the normalized or adjusted variants of the MI is fully aligned with the lattice of partitions in the above sense. Beside enhancing intuitiveness, these properties could possibly improve the efficiency of algorithms, for example, search algorithms, in the space of clusterings, though there seems not to be yet an experimental study to support such claim, calling for interesting further investigation. Nevertheless, we note that for a particular application, not always every desired property is concurrently needed at once. For example, when performing search in the space of clusterings, normalization might not be necessary, and the VI, which aligns better with the lattice of partitions, might be a more appropriate choice.

Wu et al. (2009) considered clustering comparison measures with respect to their sensitivity with class distribution. They showed that real life data can possess highly skewed class distributions, whereas some algorithms, such as K-means, tend to create balanced clusters. A good measure should therefore be sensitive to the difference in class distribution. To demonstrate this property, they used the example in Table 5, with a ground-truth clustering $\mathbf{U}$ having class sizes of [30, 2, 6, 10, 2], and two clustering solutions: $\mathbf{V}$ having cluster sizes of [10, 10, 10, 10, 10]; and $\mathbf{V}'$ having cluster sizes of [29, 2, 6, 11, 2]. It is easily seen that $\mathbf{V}'$ closely reflects the class structure in $\mathbf{U}$, and thus should be judged closer to $\mathbf{U}$ than $\mathbf{V}$. The authors showed that the unnormalized MI is a "defective measure", in that it judges $\mathrm{MI}(\mathbf{U},\mathbf{V}) > \mathrm{MI}(\mathbf{U},\mathbf{V}')$, and suggested using the "normalized VI" (d_{sum}). It can be shown that among the normalized and adjusted variants of the MI considered in this paper, only the $\mathrm{NMI}_{min}, D_{min}, d_{min}$ and Ad_{min} are defective measures in the above sense.

I	U_1	U_2	U_3	U_4	U_5	II	U_1	U_2	U_3	U_4	U_5
V_1	10	0	0	0	0	V'_1	27	0	0	2	0
V_2	10	0	0	0	0	V'_2	0	2	0	0	0
V_3	10	0	0	0	0	V'_3	0	0	6	0	0
V_4	0	0	0	10	0	V'_4	3	0	0	8	0
V_5	0	2	6	0	2	V'_5	0	0	0	0	2

Table 5: Two clustering results

6. Conclusion

This paper has presented an organized study of information theoretic measures for clustering comparison. We have shown that the normalized information distance (NID) and normalized variation of information (NVI) satisfy both the normalization and the metric properties. Between the two, the NID is preferable since the tighter upper bound of the MI used for normalization allows it to better use the [0,1] range. We highlighted the importance of correcting these measures for chance agreement, especially when the number of data points is relatively small compared with the number of clusters, for example, in the case of microarray data analysis. One of the theoretical advantages of the NID over the popular adjusted Rand index is that it can be used in the non-adjusted form (when N/K is large), thus enjoying the property of being a true metric in the space of clusterings. We therefore advocate the NID as a "general purpose" measure for clustering validation, comparison and algorithm design, for it possesses concurrently several useful and important properties. Nevertheless, we note that for a particular application scenario, not always every desired property is needed concurrently, and therefore the user should prioritize the most important property. Our research systematically organizes and complements the current literature to help readers make more informed decisions.

Acknowledgments

We thank the Action Editor and the anonymous reviewers for their constructive comments. This work was partially supported by NICTA and the Australian Research Council.

Availability: Matlab code for computing the adjusted mutual information (AMI) is available from `http://ee.unsw.edu.au/~nguyenv/Software.htm`.

Appendix A. Proofs

Proof (Theorem 1) We only prove the triangle inequality as other parts are trivial. We first show that

$$H(X|Y) \leq H(X|Z) + H(Z|Y) \tag{5}$$

holds true, since $H(X|Y) \leq H(X,Z|Y) = H(X|Z,Y) + H(Z|Y) \leq H(X|Z) + H(Z|Y)$ (the last inequality holds since conditioning always decreases entropy). We now prove the main theorem. Without loss of generality, assume that $H(Y) \leq H(X)$, and therefore $H(X|Y) \geq H(Y|X)$. The proof uses (5):

- Case 1: $H(Z) \leq H(Y)$

  $$D_{max}(X,Y) = H(X|Y) \leq H(X|Z) + H(Z|Y) \leq H(X|Z) + H(Y|Z) = D_{max}(X,Z) + D_{max}(Y,Z)$$

- Case 2: $H(Y) < H(Z) \leq H(X)$

  $$D_{max}(X,Y) = H(X|Y) \leq H(X|Z) + H(Z|Y) = D_{max}(X,Z) + D_{max}(Y,Z)$$

- Case 3: $H(X) < H(Z)$

  $$D_{max}(X,Y) = H(X|Y) \leq H(X|Z) + H(Z|Y) \leq H(Z|X) + H(Z|Y) = D_{max}(X,Z) + D_{max}(Y,Z)$$

■

Proof (Theorem 3) We prove the triangle inequality. Without loss of generality, assume that $H(X) \geq H(Y)$, therefore $H(X|Y) \geq H(Y|X)$ and $NID(X,Y) = H(X|Y)/H(X)$. The proof uses inequality (5) and simple algebra manipulations:

- Case 1: $H(Z) \leq H(Y) \leq H(X)$

  $$NID(X,Y) = \frac{H(X|Y)}{H(X)} \leq \frac{H(X|Z) + H(Z|Y)}{H(X)} \leq \frac{H(X|Z) + H(Y|Z)}{H(X)} \leq \frac{H(X|Z)}{H(X)} + \frac{H(Y|Z)}{H(Y)}$$

- Case 2: $H(Y) \leq H(Z) \leq H(X)$

  $$NID(X,Y) = \frac{H(X|Y)}{H(X)} \leq \frac{H(X|Z)}{H(X)} + \frac{H(Z|Y)}{H(X)} \leq \frac{H(X|Z)}{H(X)} + \frac{H(Z|Y)}{H(Z)} = NID(X,Z) + NID(Z,Y)$$

- Case 3: $H(Z) > H(X)$

  $$NID(X,Y) = \frac{H(X|Y)}{H(X)} < \frac{H(X|Z) + H(Z|Y)}{H(X)}$$

  If the *r.h.s* ≤ 1 then adding $H(Z) - H(X) > 0$ to both its nominator and denominator will increase it:

  $$r.h.s \leq \frac{H(X|Z) + H(Z|Y) + H(Z) - H(X)}{H(X) + H(Z) - H(X)} = \frac{H(Z|X)}{H(Z)} + \frac{H(Z|Y)}{H(Z)} = NID(X,Z) + NID(Z,Y),$$

therefore the triangle inequality holds. Otherwise if the *r.h.s* > 1 then adding $H(Z)-H(X) > 0$ to both its nominator and denominator as above will decrease it, but it will still be larger than 1. Therefore we also have:

$$NID(X,Y) \leq 1 < \frac{H(X|Z)+H(Z|Y)+H(Z)-H(X)}{H(X)+H(Z)-H(X)} = NID(X,Z)+NID(Z,Y).$$

■

Proof (Theorem 4) Again only the triangle inequality proof is of interest. It is sufficient to prove the following inequality:

$$\frac{H(X|Y)}{H(X,Y)} \leq \frac{H(X|Z)}{H(X,Z)} + \frac{H(Z|Y)}{H(Z,Y)},$$

then swap X and Y to obtain another analogous inequality and add them together we have the triangle inequality. The proof uses inequality (5) and simple algebra manipulations:

$$\frac{H(X|Y)}{H(X,Y)} = \frac{H(X|Y)}{H(Y)+H(X|Y)} \leq \frac{H(X|Z)+H(Z|Y)}{H(Y)+H(X|Z)+H(Z|Y)} = \frac{H(X|Z)+H(Z|Y)}{H(X|Z)+H(Y,Z)} = \dots$$
$$= \frac{H(X|Z)}{H(X|Z)+H(Y,Z)} + \frac{H(Z|Y)}{H(X|Z)+H(Y,Z)} \leq \frac{H(X|Z)}{H(X|Z)+H(Z)} + \frac{H(Z|Y)}{H(Y,Z)} = \frac{H(X|Z)}{H(X,Z)} + \frac{H(Z|Y)}{H(Z,Y)}.$$

■

Proof (Theorems 2 and 5) It is sufficient to point out a single counter example where the triangle inequality is violated. Let X and Y be two *independent* random binary variables with probability $P(X=1)=P(X=0)=P(Y=1)=P(Y=0)=1/2$, then $Z=[X;Y]$ is also a random variable with four discrete values. It is straightforward to verify that the triangle inequality is violated for all the mentioned distance measures, for example, $D_{min}(X,Y)=1 < D_{min}(X,Z)+D_{min}(Y,Z)=0$. ■

Proof (Theorem 6) For $N=5$, a counter example for the triangle inequality is the following three clusterings: $\mathbf{U}=\{U_3U_1U_1U_1U_2\}, \mathbf{V}=\{V_2V_2V_3V_1V_2\}, \mathbf{X}=\{X_2X_1X_1X_1X_2\}$.

Similarly, for $N=5+d$ $(d \in \mathbb{N}^+)$, a counter example for the triangle inequality is the following three clusterings: $\mathbf{U}=\{U_3U_1U_1U_1U_2U_6U_7\dots U_{5+d}\}$, $\mathbf{V}=\{V_2V_2V_3V_1V_2V_6V_7\dots V_{5+d}\}$, $\mathbf{X}=\{X_2X_1X_1X_1X_2X_6X_7\dots X_{5+d}\}$. ■

Proof (Theorem 7) The following facts from the generalized hypergeometric distribution will be useful:

$$\mathbf{E}(n_{ij}) = \sum_{n_{ij}} n_{ij}\mathcal{P}(M|n_{ij},a,b) = \frac{a_i,b_j}{N}, \tag{6}$$
$$\mathbf{E}(n_{ij}^2) = \sum_{n_{ij}} n_{ij}^2\mathcal{P}(M|n_{ij},a,b) = \frac{a_i(a_i-1)b_j(b_j-1)}{N(N-1)} + \frac{a_ib_j}{N},$$

where $\mathcal{P}\{M|n_{ij},a,b\} = \frac{\binom{N}{n_{ij}}\binom{N-n_{ij}}{a_i-n_{ij}}\binom{N-a_i}{b_j-n_{ij}}}{\binom{N}{a_i}\binom{N}{b_j}}$ is the probability of encountering a contingency table M having fixed marginals a,b and the $(i,j)-th$ entry being n_{ij} under the generalized hypergeometric

model of randomness. Note that for the sake of notational simplicity we have dropped the lower and upper values of n_{ij} which runs from $\max((a_i+b_j-N),0)$ to $\min(a_i,b_j)$ in the sums. From (6) we have:

$$\mathbf{E}(n_{ij}) = \sum_{n_{ij}} n_{ij}\mathcal{P}(M|n_{ij},a,b) = \frac{a_i b_j}{N}\sum_{n_{ij}} \frac{n_{ij}N}{a_i b_j}\mathcal{P}(M|n_{ij},a,b) = \frac{a_i b_j}{N},$$

therefore: $\sum_{n_{ij}} \frac{n_{ij}N}{a_i b_j}\mathcal{P}(M|n_{ij},a,b) = 1$. Let $Q(n_{ij}) = \frac{n_{ij}N}{a_i b_j}\mathcal{P}(M|n_{ij},a,b)$, then we can think of $Q(n_{ij})$ as a discrete probability distribution on n_{ij}. Applying Jensen's inequality ($\mathbf{E}(f(x)) \leq f(\mathbf{E}(x))$ for f concave) to the concave logarithm function yields:

$$\sum_{n_{ij}} \frac{n_{ij}}{N}\log(\frac{N.n_{ij}}{a_i b_j})\mathcal{P}(M|n_{ij},a,b) = \sum_{n_{ij}} \frac{a_i b_j}{N^2}\log(\frac{N.n_{ij}}{a_i b_j})Q(n_{ij}) \leq \frac{a_i b_j}{N^2}\log\left(\mathbf{E}_Q(\frac{N.n_{ij}}{a_i b_j})\right). \quad (7)$$

Now, let us calculate $\mathbf{E}_Q(\frac{N.n_{ij}}{a_i b_j})$:

$$\begin{aligned}\mathbf{E}_Q(\frac{N.n_{ij}}{a_i b_j}) &= \sum_{n_{ij}} \frac{N.n_{ij}}{a_i b_j}Q(n_{ij}) = \sum_{n_{ij}} \frac{N.n_{ij}}{a_i b_j}\frac{n_{ij}N}{a_i b_j}\mathcal{P}(M|n_{ij},a,b) = \frac{N^2}{a_i^2 b_j^2}\sum_{n_{ij}} n_{ij}^2\mathcal{P}(M|n_{ij},a,b) \\ &= \frac{N^2}{a_i^2 b_j^2}\left(\frac{a_i(a_i-1)b_j(b_j-1)}{N(N-1)} + \frac{a_i b_j}{N}\right) = \frac{N(a_i-1)(b_j-1)}{(N-1)a_i b_j} + \frac{N}{a_i b_j}.\end{aligned}$$

Substituting this expression into (7) yields:

$$\sum_{n_{ij}} \frac{n_{ij}}{N}\log(\frac{N.n_{ij}}{a_i b_j})\mathcal{P}(M|n_{ij},a,b) \leq \frac{a_i b_j}{N^2}\log\left(\frac{N(a_i-1)(b_j-1)}{(N-1)a_i b_j} + \frac{N}{a_i b_j}\right).$$

Finally:

$$\mathbf{E}\{I(\mathbf{U},\mathbf{V})\} = \sum_{i=1}^{R}\sum_{j=1}^{C}\sum_{n_{ij}} \frac{n_{ij}}{N}\log(\frac{N.n_{ij}}{a_i b_j})P(M|n_{ij},a,b) \leq \sum_{i=1}^{R}\sum_{j=1}^{C} \frac{a_i b_j}{N^2}\log\left(\frac{N(a_i-1)(b_j-1)}{(N-1)a_i b_j} + \frac{N}{a_i b_j}\right). \quad (8)$$

Note that $\sum_{ij} a_i b_j/N^2 = 1$, continue applying Jensen's inequality on (8) yields:

$$\mathbf{E}\{I(\mathbf{U},\mathbf{V})\} \leq \log\left(\sum_{i=1}^{R}\sum_{j=1}^{C} \frac{a_i b_j}{N^2}(\frac{N(a_i-1)(b_j-1)}{(N-1)a_i b_j} + \frac{N}{a_i b_j})\right) = \log\left(\frac{N+RC-R-C}{N-1}\right)$$

■

References

A. N. Albatineh, M. Niewiadomska-Bugaj, and D. Mihalko. On similarity indices and correction for chance agreement. *Journal of Classification*, 23(2):301–313, 2006.

S. Asur, D. Ucar, and S. Parthasarathy. An ensemble framework for clustering protein-protein interaction networks. *Bioinformatics*, 23(13):i29–i40, 2007.

A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *J. Mach. Learn. Res.*, 6:1345–1382, 2005.

S. Ben-David, U. von Luxburg, and D. Pal. A sober look at clustering stability. In *19th Annual Conference on Learning Theory (COLT 2006)*, pages 5–19, 2006.

M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *FOCS '03: Procs. IEEE Symposium on Foundations of Computer Science*, 2003.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

B. E. Dom. An information-theoretic external cluster-validity measure. Technical report, Research Report RJ 10219, IBM, 2001.

X. Z. Fern and C. E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Procs. ICML'03*, pages 186–193, 2003.

Z. He, X. Xu, and S. Deng. k-anmi: A mutual information based clustering algorithm for categorical data. *Inf. Fusion*, 9(2):223–233, 2008.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classif.*, 2(1):193–218, 1985.

A. Kraskov, H. Stogbauer, R. G. Andrzejak, and P. Grassberger. Hierarchical clustering using mutual information. *EPL (Europhysics Letters)*, 70(2):278–284, 2005.

T. O. Kvalseth. Entropy and correlation: Some comments. *Systems, Man and Cybernetics, IEEE Transactions on*, 17(3):517–519, 1987.

H.O Lancaster. The chi-squared distribution. New York, 1969. John Wiley.

M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi. The similarity metric. *Information Theory, IEEE Transactions on*, 50(12):3250–3264, 2004.

Z. Liu, Z. Guo, and M. Tan. Constructing tumor progression pathways and biomarker discovery with fuzzy kernel kmeans and dna methylation data. *Cancer Inform*, 6:1–7, 2008.

P. Luo, H. Xiong, G. Zhan, J. Wu, and Z. Shi. Information-theoretic distance measures for clustering validation: Generalization and normalization. *IEEE Trans. on Knowl. and Data Eng.*, 21(9): 1249–1262, 2009.

M. Meilă. Comparing clusterings by the variation of information. In *COLT '03*, pages 173–187, 2003.

M. Meilă. Comparing clusterings: an axiomatic view. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 577–584, 2005. ISBN 1-59593-180-5.

M. Meilă. Comparing clusterings—an information based distance. *J. Multivar. Anal.*, 98(5):873–895, 2007.

S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Mach. Learn.*, 52(1-2): 91–118, 2003.

W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

O. Shamir and N. Tishby. Model selection and stability in k-means clustering. In *21th Annual Conference on Learning Theory (COLT 2008)*, 2008.

V. Singh, L. Mukherjee, J. Peng, and J. Xu. Ensemble clustering using semidefinite programming with applications. *Mach. Learn.*, 2009. doi: 10.1007/s10994-009-5158-y.

D. Steinley. Properties of the Hubert-Arabie adjusted Rand index. *Psychol Methods*, 9(3):386–96, 2004.

A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

K. Tumer and A.K. Agogino. Ensemble clustering with voting active clusters. *Pattern Recognition Letters*, 29(14):1947–1953, 2008.

N. X. Vinh and J. Epps. A novel approach for automatic number of clusters detection in microarray data based on consensus clustering. In *BIBE'09: Procs. IEEE Int. Conf. on BioInformatics and BioEngineering*, 2009.

N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *ICML '09*, 2009.

M. Warrens. On similarity coefficients for 2x2 tables and correction for chance. *Psychometrika*, 73 (3):487–502, 2008.

J. Wu, H. Xiong, and J. Chen. Adapting the right measures for k-means clustering. In *KDD '09*, 2009.

Y. Y. Yao. Information-theoretic measures for knowledge discovery and data mining. In *Entropy Measures, Maximum Entropy Principle and Emerging Applications*, pages 115–136. Karmeshu (ed.), Springer, 2003.

Z. Yu, H-S. Wong, and H. Wang. Graph-based consensus clustering for class discovery from gene expression data. *Bioinformatics*, 23(21):2888–2896, 2007.

Journal of Machine Learning Research 11 (2010) 2855-2900 Submitted 6/09; Revised 8/10; Published 10/10

Expectation Truncation and the Benefits of Preselection In Training Generative Models

Jörg Lücke LUECKE@FIAS.UNI-FRANKFURT.DE
Frankfurt Institute for Advanced Studies (FIAS)
Goethe-Universität Frankfurt
Ruth-Moufang-Str. 1
60438 Frankfurt am Main
Germany

Julian Eggert JULIAN.EGGERT@HONDA-RI.DE
Honda Research Institute Europe
Carl-Legien-Str. 30
63073 Offenbach am Main
Germany

Editor: Aapo Hyvärinen

Abstract

We show how a preselection of hidden variables can be used to efficiently train generative models with binary hidden variables. The approach is based on Expectation Maximization (EM) and uses an efficiently computable approximation to the sufficient statistics of a given model. The computational cost to compute the sufficient statistics is strongly reduced by selecting, for each data point, the relevant hidden causes. The approximation is applicable to a wide range of generative models and provides an interpretation of the benefits of preselection in terms of a variational EM approximation. To empirically show that the method maximizes the data likelihood, it is applied to different types of generative models including: a version of non-negative matrix factorization (NMF), a model for non-linear component extraction (MCA), and a linear generative model similar to sparse coding. The derived algorithms are applied to both artificial and realistic data, and are compared to other models in the literature. We find that the training scheme can reduce computational costs by orders of magnitude and allows for a reliable extraction of hidden causes.

Keywords: maximum likelihood, deterministic approximations, variational EM, generative models, component extraction, multiple-cause models

1. Introduction

In many applications of artificial and biological systems, data interpretation is challenging because of noise, the complexity of the input and because of its ambiguity. Optimal inference based on probabilistic generative models is in general intractable in such situations because it involves the evaluation of all potential interpretations of the input. To approximate optimal inference, a fast initial stage of processing has therefore long since been suggested. In applications to visual data, a first processing stage can select candidate objects or components that are potential causes of the given input (see, e.g., Yuille and Kersten, 2006). Based on these candidates the meaning of a data point is subsequently inferred in a second recurrent stage. The strategy of candidate preselection has indeed been suggested and applied in different contexts (e.g., Körner et al., 1999; Lee and

Mumford, 2003; Yuille and Kersten, 2006; Westphal and Würtz, 2009) and prominent systems of feed-forward processing (such as Riesenhuber and Poggio, 1999, and van Rullen and Thorpe, 2001) are sometimes interpreted as sophisticated preprocessing stages for a subsequent recurrent stage. The general idea of candidate preselection followed by recurrent recognition will in this paper be formulated in terms of a variational approximation that allows for an efficient training of probabilistic generative models.

In Section 2 the approximation scheme will be introduced as an approximation to Expectation Maximization (EM). In Section 3 we systematically derive it as a variational EM approach. In Section 4 the training scheme is applied to a number of different generative models and a number of different data types. Section 5 discusses the features of the novel approach and the obtained results.

2. Expectation Maximization and Expectation Truncation

Our approach is based on Expectation Maximization (EM; Dempster et al., 1977) which is used to maximize the data likelihood under a given generative model:

$$\Theta^* \;=\; \text{argmax}_\Theta\{L(\Theta)\} \quad \text{with} \quad L(\Theta) = \log\left(p(\vec{y}^{\,(1)},\ldots,\vec{y}^{\,(N)}\,|\,\Theta)\right), \tag{1}$$

where Θ are the parameters of a given generative model and where the N data points, $\{\vec{y}^{\,(n)}\}_{n=1,\ldots,N}$, will be taken to be generated independently from a stationary process.

To find the parameters Θ^* at least approximately, we use the EM approach as it was formalized, for example, by Neal and Hinton (1998) and introduce the free-energy function $\mathcal{F}(q,\Theta)$ which is a function of Θ and an unknown distribution $q(\vec{s}^{\,(1)},\ldots,\vec{s}^{\,(N)})$ over the hidden variables. $\mathcal{F}(q,\Theta)$ can be shown to be a lower bound of the likelihood evaluated at the same parameter values. For our purposes we assume independently generated data vectors $\vec{y}^{\,(n)}$ and use (without loss of generality) a distribution q which is factored over the data points, $q(\vec{s}^{\,(1)},\ldots,\vec{s}^{\,(N)}) = \prod_n q^{(n)}(\vec{s}^{\,(n)};\Theta^{\text{old}})$. Note that we take q to be parameterized by Θ^{old}. The free-energy can thus be written as:

$$\mathcal{F}(q,\Theta) = \sum_{n=1}^{N}\left[\sum_{\vec{s}} q^{(n)}(\vec{s};\Theta^{\text{old}})\;\left[\log\left(p(\vec{y}^{\,(n)}\,|\,\vec{s},\Theta)\right) + \log\left(p(\vec{s}|\,\Theta)\right)\right]\right] + H(q)\,, \tag{2}$$

where $H(q) = -\sum_n \sum_{\vec{s}} q^{(n)}(\vec{s};\Theta^{\text{old}})\log(q^{(n)}(\vec{s};\Theta^{\text{old}}))$ is a function (the Shannon entropy) that is independent of Θ. The sum over all states of $\vec{s}$ becomes an integral if the values of $\vec{s}$ are continuous. In the EM scheme $\mathcal{F}(q,\Theta)$ is maximized alternately with respect to q in the E-step (while Θ is kept fixed) and with respect to Θ in the M-step (while q is kept fixed). It can be shown that the EM iterations increase the likelihood or keep it constant. In practical applications EM is found to increase the likelihood to (at least local) likelihood maxima.

The free-energy function (2) can be used to derive update rules for the parameters Θ of a given model. Such a derivation can in some cases be challenging but one usually arrives at expressions in which the new set of parameters Θ is a function of the old set Θ^{old} and the data $\{\vec{y}^{\,(n)}\}_{n=1,\ldots,N}$. The update rules derived contain what is often referred to as the *sufficient statistics* of the model, that is, they contain expressions of the form

$$\langle g(\vec{s})\rangle_{q^{(n)}} = \sum_{\vec{s}} q^{(n)}(\vec{s};\Theta^{\text{old}})\, g(\vec{s})\,, \tag{3}$$

where $g(\vec{s})$ is a function of the hidden variables. The functions $g(\vec{s})$ are often relatively simple, for example, $g(\vec{s}) = s_i$ or $g(\vec{s}) = s_i s_j$, but can for some models be more elaborate and may also include

parameter dependencies. For an exact E-step in the EM scheme the functions $q^{(n)}(\vec{s};\Theta^{\text{old}})$ are given by

$$q^{(n)}(\vec{s};\Theta^{\text{old}}) = p(\vec{s}|\vec{y}^{(n)},\Theta^{\text{old}}) = \frac{p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^{\text{old}})}{\sum_{\vec{s}'} p(\vec{s}',\vec{y}^{(n)}\,|\,\Theta^{\text{old}})}, \tag{4}$$

where $p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^{\text{old}}) = p(\vec{s}|\,\Theta^{\text{old}})\,p(\vec{y}^{(n)}\,|\,\vec{s},\Theta^{\text{old}})$ with the latter distributions being defined by the used generative model. To train models with multiple causes, the computation of the exact sufficient statistics is usually avoided because it involves summing or integrating over a large space of hidden states in (3) and (4). To reduce computational costs, training schemes therefore use approximations to these intractable sums (or integrals) or approximations to the exact posterior $p(\vec{s}|\vec{y}^{(n)},\Theta^{\text{old}})$. The approximation method discussed in this paper will be introduced as an approximation to the sufficient statistics.

Let us consider the sufficient statistics of a function g given by the combination of (3) and (4):

$$\langle g(\vec{s})\rangle_{q^{(n)}} = \frac{\sum_{\vec{s}} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^{\text{old}})\; g(\vec{s})}{\sum_{\vec{s}'} p(\vec{s}',\vec{y}^{(n)}\,|\,\Theta^{\text{old}})}. \tag{5}$$

Again, we have to sum over a very large space of hidden states. Let us for instance assume that we have already found the optimal or approximately optimal parameters $\Theta^{\text{old}} \approx \Theta^{*}$, that is, let us assume that any given input vector is well represented by a distribution over hidden states. A given $\vec{y}^{(n)}$ is in this case usually well represented by a distribution over just a small set of hidden vectors. For the sums in (5) this means that just some summands contribute significantly while the others are negligible. Thus, if we could find the right summands for a given $\vec{y}^{(n)}$, we could expect a good approximation of $\langle g(\vec{s})\rangle_{q^{(n)}}$ in (5) without having to sum over the entire state space of $\vec{s}$.

More formally, if $\mathcal{K}_n$ denotes the set of all states that contain significant contributions to the sums in (5), it applies:

$$\langle g(\vec{s})\rangle_{q^{(n)}} \approx \frac{\sum_{\vec{s}\in\mathcal{K}_n} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^{\text{old}})\; g(\vec{s})}{\sum_{\vec{s}'\in\mathcal{K}_n} p(\vec{s}',\vec{y}^{(n)}\,|\,\Theta^{\text{old}})}. \tag{6}$$

A potential subset containing the relevant summands could be found by exploiting specific data properties. If the data was generated by few hidden units on average, for instance, most data points are well approximated by only considering combinations of few active causes. A subset for the approximation in (6) for binary causes s_j could thus be given by $\mathcal{K} = \{\vec{s}\;|\;\sum_j s_j \leq \gamma\}$, where γ is the maximal number of active causes. Such a choice can significantly reduce the number of states that have to be evaluated. Depending on γ the combinatorics can still be considerable, however, and still, just a few of the summands might contribute.

To further constrain the state space, let us suppose that we can in some way find functions $\mathcal{S}_h : \mathbb{R}^D \to \mathbb{R}$ that give estimates of how likely it is for the hidden causes $h = 1,\ldots,H$ to have contributed to the generation of a specific input $\vec{y}^{(n)}$. If we had such functions, we could approximate the sufficient statistics (5) by neglecting all causes that are unlikely to have contributed. In other words, we could just sum over a subset $\mathcal{K}_n \subseteq \mathcal{K}$ that contains the combinations of all hidden variables that are likely to have caused the input $\vec{y}^{(n)}$. To define such a set $\mathcal{K}_n$ more formally,

consider the $h = 1,\ldots,H$ values $\mathcal{S}_h(\vec{y}^{(n)})$ for a given data point $\vec{y}^{(n)}$. To select H' candidates, we define the index set I to contain those latent indices h with the H' largest values $\mathcal{S}_h(\vec{y}^{(n)})$. The set $\mathcal{K}_n$ is then given by:

$$\mathcal{K}_n = \{\vec{s} \mid \textstyle\sum_j s_j \leq \gamma \text{ and } (\forall i \notin I : s_i = 0)\}. \tag{7}$$

Equation 6 represents a good approximation if the contributions of the states not in $\mathcal{K}_n$ are indeed negligible compared to the contributions of the states in $\mathcal{K}_n$. Much depends, of course, on the functions $\mathcal{S}_h$. From these functions, which we will term *selection functions*, we demand the following properties: (A) they have to be efficiently computable and (B) they have to give high values for hidden variables that actually are responsible for a given $\vec{y}^{(n)}$. Note that for the selection functions it is merely important to select candidates that can *potentially* explain the input. Neither are their exact values used in (6) nor is it disadvantageous for the accuracy of the approximation if some candidates are selected that turn out to contribute very little. We will see examples of such selection functions for different generative models in Section 4. Before let us summarize the approximation discussed above in the form of the pseudo-code given in Algorithm 1. As the approximation scheme resides on a truncation of the sums in the expectation value computations in (5), we will refer to it as *Expectation Truncation* (ET).

Algorithm 1: Expectation Truncation - Pseudo Code

```
Choose approximation parameters H' and γ (γ ≤ H' ≤ H) and randomly initialize the
parameters of the generative model.
while parameters have not converged do
    for all data points n = 1,...,N do
        Compute the selection function value S_h for each h = 1,...,H and determine the
        index set I of the H' hidden variables with the H' highest values for S_h.
        Compute the set of binary vectors K_n = {s | Σ_j s_j ≤ γ and (∀ i ∉ I : s_i = 0)}
        Compute the approximate sufficient statistics (6).
    Update the parameters in the M-step using the approximate sufficient statistics.
```

The two parameters H' and γ control the size of $\mathcal{K}_n$. H' determines how many candidates are selected and γ fixes the maximal number of non-zero hidden units s_h. For instance, if we choose $H' = 4$ and $\gamma = 2$, the summation over $\vec{s}$ considers four candidates of which either none, one, or two are simultaneously active (compare Figure 2). The size of $\mathcal{K}_n$ is thus given by $\sum_{\gamma'=0}^{\gamma} \binom{H'}{\gamma'}$. The approximation's accuracy increases with increasing values of H' and γ but its computational demand increases as well. For the highest possible values, $H' = \gamma = H$, we drop, for any selection function $\mathcal{S}_h$, back to the case of the exact sufficient statistics (5).

Note that while the approximation scheme presumably results in good approximations for many data points it can be expected to give poor results for data points generated by more than γ hidden causes (i.e., for data points not in $\mathcal{K}$). To avoid learning from such data points with inappropriately estimated sufficient statistics let us again assume that we have already found close to optimal parameters Θ^*. In such a situation and for any data point $\vec{y}^{(n)}$ generated by less or equal γ hidden

causes we get:

$$p(\vec{y}^{(n)}\,|\,\Theta^*) = \sum_{\vec{s}} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^*) \approx \sum_{\vec{s}\in\mathcal{K}_n} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^*)\,, \tag{8}$$

where we have assumed appropriate selection functions and relatively low noise levels. In the same situation but for data points $\vec{y}^{(n)}$ which were generated by more than γ hidden causes we obtain:

$$p(\vec{y}^{(n)}\,|\,\Theta^*) = \sum_{\vec{s}} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^*) \approx \sum_{\vec{s}\notin\mathcal{K}_n} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^*) \;\;\Rightarrow\;\; \sum_{\vec{s}\in\mathcal{K}_n} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^*) \approx 0\,. \tag{9}$$

The values of the sums $\sum_{\vec{s}\in\mathcal{K}_n} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^*)$ for the different data points can thus serve as indicators for finding data points that were presumably generated by less than γ causes. For learning we aim to include only those data points that are approximated well. We therefore define a set $\mathcal{M}$ of the $N^{\text{cut}} \leq N$ data points with largest sums. In the beginning of learning, the estimation of such data points is too imprecise, however. We therefore start by taking all data points into account $N^{\text{cut}} = N$ and decrease N^{cut} to values close to $N^{\leq\gamma}$ within the last third of the iterations. $N^{\leq\gamma}$ is hereby the expected number of data points generated by less or equal γ causes. For a given generative model, this number can usually be computed tractably. The number of data points generated by $\leq\gamma$ causes can, for a given data set, be smaller than $N^{\leq\gamma}$ because of finitely many data points. It can therefore be beneficial to finally use an N^{cut} slightly smaller than $N^{\leq\gamma}$. This potentially avoids the consideration of data points that are not well approximated. In numerical experiments in Section 4 we will therefore use final values of $N^{\text{cut}} = 0.9\,N^{\leq\gamma}$ although $N^{\text{cut}} = N^{\leq\gamma}$ gives similar results especially for large N.

Considering Algorithm 1 what is still left to specify are concrete expressions for the selection functions $\mathcal{S}_h$ and expressions for parameter update rules (M-step equations). These equations do, however, depend on the particular generative model the method is applied to. We will therefore discuss selection functions and update rules individually for the different generative models we investigate in Section 4. Given selection functions and update rules, Algorithm 1 describes an approximation scheme applicable to generative models with binary hidden variables. The scheme has been introduced and defined as an approximation to the sufficient statistics (5). In the next section it will be systematically derived as a variational EM approach. The assumptions used for the approximation will thus become explicit, allowing a generalization of the scheme and a specification of its potential limitations. The computational complexity of the method will be discussed in Appendix C.

3. Expectation Truncation and Variational EM

In this section we will show that Expectation Truncation corresponds to a variational EM approximation. The approximation, as introduced in Section 2, consists of two parts: (A) an approximation to the sufficient statistics in Equation 6, and (B) a selection of data points that are well approximated by Equation 6. Both of these parts can be formulated as variational EM steps. For the derivation we will start with (B), that is, with the selection of data points. The selection will take the form of a classification of data points into two classes: one class that contains the data points that can be well approximated, and another class that contains the remaining points. The corresponding variational step will be referred to as the first variational step (Section 3.1). The step (A) corresponding to approximation (6) will only be the second variational step (Section 3.2). Both steps combine to form the approximation scheme of Expectation Truncation. The scheme is compactly summarized in Section 3.3, and the basic procedural steps are listed in Algorithm 2 which represents a generalization of Algorithm 1.

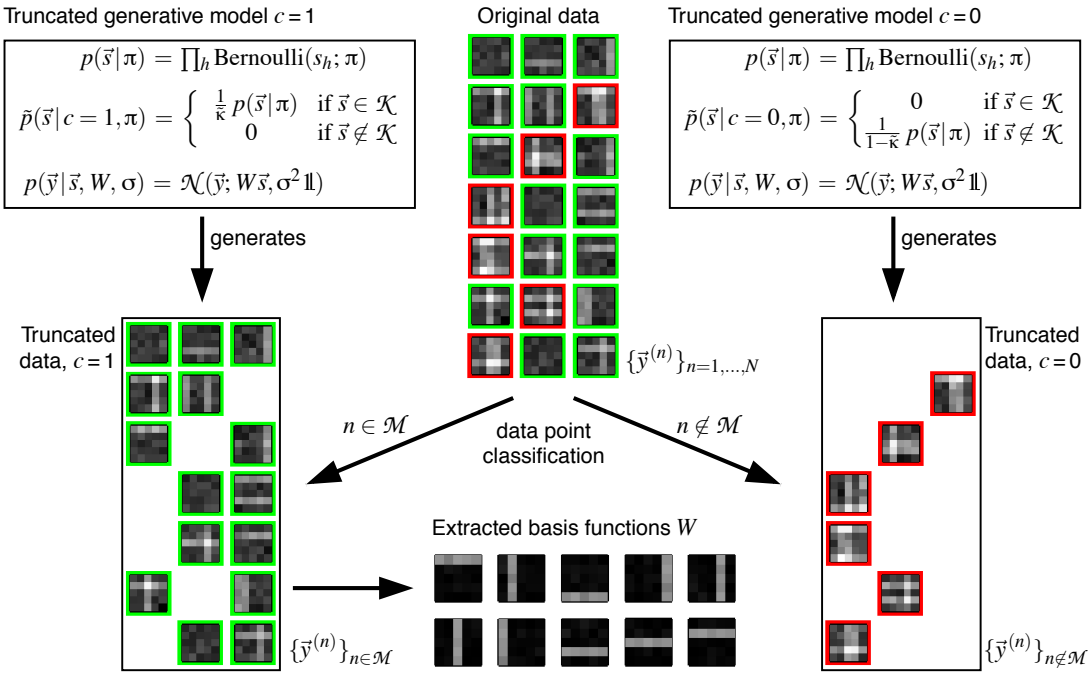


Figure 1: First variational approximation: data classification. The figure shows Expectation Truncation (without preselection) for a concrete generative model. In this example, the generative model generates data by linearly superimposing basis functions in the form of horizontal and vertical bars. Data generated by the original model contains up to ten bars chosen with a Bernoulli prior (example data points are shown in the center). Data generated by the truncated generative model with $c = 1$ contains up to two bars (we set $\mathcal{K} = \{\vec{s} \mid \sum_j s_j \leq \gamma\}$ with $\gamma = 2$). Data generated by the truncated generative model with $c = 0$ contains at least three bars. If we train the truncated generative model with $c = 1$ on data from which data points with $\sum_j s_j > \gamma$ were removed, we can expect to approximately recover the true generating basis functions W of the original model.

3.1 First Variational Approximation: Data Classification

As in Section 2, let us consider a generative model with a set of hidden variables denoted by $\vec{s}$, a set of observed variables denoted by $\vec{y}$, and a set of parameters denoted by Θ. Let us denote the prior distribution of the model by the (not further specified) function $p(\vec{s}|\Theta)$, and the noise distribution by the (not further specified) function $p(\vec{y}|\vec{s},\Theta)$. To distinguish this generative model from models introduced later, it will from now on be referred to as the *original* generative model.

We will formalize the classification of data points by introducing two new generative models defined based on the original model. The two models will correspond to two classes of data points: one class of those points that can be well approximated, and one class of points that can not. Let $\mathcal{K}$ be a subset of the space of all possible values of $\vec{s}$. Given such a set, we define the two generative

models by introducing two new prior distributions that are based on the original prior:

$$p(\vec{s}\,|\,c=1,\Theta) = \begin{cases} \frac{1}{\tilde{\kappa}}\,p(\vec{s}\,|\,\Theta) & \text{if } \vec{s}\in\mathcal{K} \\ 0 & \text{if } \vec{s}\notin\mathcal{K} \end{cases} \text{ and } p(\vec{s}\,|\,c=0,\Theta) = \begin{cases} 0 & \text{if } \vec{s}\in\mathcal{K} \\ \frac{1}{1-\tilde{\kappa}}\,p(\vec{s}\,|\,\Theta) & \text{if } \vec{s}\notin\mathcal{K} \end{cases} \tag{10}$$

where $\tilde{\kappa} = \sum_{\vec{s}\in\mathcal{K}} p(\vec{s}\,|\,\Theta)$. We take the noise distribution $p(\vec{y}\,|\,\vec{s},\Theta)$ of the new models to be identical to the original noise distribution. We will refer to the new generative models as *truncated* models because their prior distributions are truncated to be zero outside of specific subsets. Note that the generation of data according to the truncated model with $c=1$ corresponds to generating data according to the original model while only accepting data points generated by $\vec{s}\in\mathcal{K}$. Analogously, generating data according to the truncated model with $c=0$ is equivalent to generating data according to the original model while accepting only data points generated by $\vec{s}\notin\mathcal{K}$. Figure 1 shows the truncated generative models and data they generate for a concrete example (a model that linearly combines bar-like generative fields; compare Section 4.1). For the example, $\mathcal{K}$ is the set of binary states $\vec{s}$ with less or equal γ non-zero entries. In general, $\mathcal{K}$ can be any subset, however.

Let us now mix the two truncated models in Equation 10 by introducing $c\in\{0,1\}$ as additional hidden variable and by drawing $c=1$ with probability κ. The prior distribution of this mixed model is thus given by:

$$p(c\,|\,\kappa) \;=\; \kappa^{c}\,(1-\kappa)^{1-c}\,, \tag{11}$$

$$p(\vec{s}\,|\,c,\Theta) \;=\; \Big(\frac{c}{\tilde{\kappa}}\;\delta(\vec{s}\in\mathcal{K}) + \frac{1-c}{1-\tilde{\kappa}}\;\delta(\vec{s}\notin\mathcal{K})\Big)\,p(\vec{s}\,|\,\Theta)\,. \tag{12}$$

where we have introduced $\delta(\vec{s}\in\mathcal{K})=1$ if $\vec{s}\in\mathcal{K}$ and zero otherwise, and $\delta(\vec{s}\notin\mathcal{K})=1$ if $\vec{s}\notin\mathcal{K}$ and zero otherwise. We will refer to this model as the *mixed* generative model. Note that the mixed model is identical to the original generative model if we choose $\kappa=\tilde{\kappa}=\sum_{\vec{s}\in\mathcal{K}} p(\vec{s}\,|\,\Theta)$ as mixing proportion. The mixed model thus contains the original model as a special case.

Now, consider a set of N data points $\{\vec{y}^{\,(n)}\}_{n=1,\ldots,N}$ generated according to the original generative model. Let us maximize the likelihood of the data under the mixed model (11) and (12). If we use EM for optimization (compare Section 2), we obtain the free-energy

$$\begin{aligned}\tilde{\mathcal{F}}\,(q,\Theta,\kappa) \;\;=\;\; & \sum_{n}\sum_{c} q^{(n)}(c\,;\Theta^{\text{old}})\,\log\big(p(\vec{y}^{\,(n)}\,|\,c,\Theta)\big) \\ & +\log(\kappa)\sum_{n} q^{(n)}(c=1;\Theta^{\text{old}}) + \log(1-\kappa)\sum_{n} q^{(n)}(c=0;\Theta^{\text{old}}) + H(q)\,,\end{aligned} \tag{13}$$

where $H(q)$ is the entropy w.r.t. $q^{(n)}(c;\Theta^{\text{old}})$ (summed over all n and c). The free-energy (13) can be optimized iteratively by maximizing q in the E-step and (Θ,κ) in the M-step. For the E-step, choosing the exact posterior, $q^{(n)}(c\,;\,\Theta^{\text{old}}) = p(c\,|\,\vec{y}^{\,(n)},\Theta^{\text{old}})$, represents the optimal choice. Unfortunately, it is computationally intractable in general because

$$p(c=1\,|\,\vec{y}^{\,(n)},\Theta^{\text{old}}) = \frac{\sum_{\vec{s}\in\mathcal{K}} p(\vec{y}^{\,(n)},\vec{s}\,|\,\Theta)}{\sum_{\vec{s}} p(\vec{y}^{\,(n)},\vec{s}\,|\,\Theta)} \tag{14}$$

requires a summation over the entire state space of $\vec{s}$ (similarly for $c=0$). We thus choose a variational approximation to the true posterior by setting $q^{(n)}(c\,;\Theta^{\text{old}})$ to zero or one. This approximation

reduces the free-energy (13) to:

$$\tilde{\mathcal{F}}(q,\Theta) \;=\; \overbrace{\sum_{n\in\mathcal{M}} \log\left(p(\vec{y}^{\,(n)}\,|\,c=1,\Theta)\right)}^{\approx L_1(\Theta)} + \overbrace{\sum_{n\notin\mathcal{M}} \log\left(p(\vec{y}^{\,(n)}\,|\,c=0,\Theta)\right)}^{\approx L_0(\Theta)} \\ +\log(\kappa)|\mathcal{M}| + \log(1-\kappa)(N-|\mathcal{M}|) + H(q)\,. \tag{15}$$

where $\mathcal{M} = \{n\,|\,q^{(n)}(c=1\,;\Theta^{\rm old})=1\}$. Note that κ can be optimized independently of Θ because the first two summands in (15) only depend on Θ. As we also know its final optimal value ($\kappa=\tilde{\kappa}$), we will treat the mixing proportion as implicitly known. κ can thus be omitted as a parameter of the free-energy (15) and will not play a role for our further considerations.

For the data set $\mathcal{M}$ note that the best choice for $q^{(n)}(c\,;\Theta^{\rm old})$ under the constraint $q^{(n)}(c\,;\Theta^{\rm old})\in\{0,1\}$ is given by the assignment $q^{(n)}(c=1\,;\Theta^{\rm old})=1$ if $\vec{y}^{\,(n)}$ was generated by class $c=1$ (and zero otherwise). This would amount to setting $\mathcal{M}$ to

$$\mathcal{M}^{\rm opt} = \{n\,|\,\vec{y}^{\,(n)} \text{ generated by class } c=1\}\,. \tag{16}$$

In general this best choice can not be computed exactly. We can, however, derive tractable approximations to $\mathcal{M}^{\rm opt}$. Choosing a set $\mathcal{M}$ is equivalent to choosing a distribution $q^{(n)}(c\,;\Theta^{\rm old})$ with binary values (as an approximation to Equation 14). Any choice of $\mathcal{M}$ thus represents a variational approximation. In Appendix A.2 it is shown that one such approximation is obtained via sorting according to the denominator values in Equation 6. The selection of $N^{\rm cut}$ data points obtained in this way is thus equivalent to a variational E-step.

An interesting aspect of Equation 15 is that the first two summands take the form of two log-likelihoods. The first summand is the likelihood $L_1(\Theta)$ of the truncated generative model with $c=1$, and the second is the likelihood $L_0(\Theta)$ of the model with $c=0$. If $\mathcal{M}=\mathcal{M}^{\rm opt}$, both likelihoods are evaluated on the set of data points they can generate (see Figure 1). Considering these properties of Equation 15 the question arises how the maximum of $\tilde{\mathcal{F}}(q,\Theta)$ and the maxima of $L_1(\Theta)$ and $L_0(\Theta)$ are related. It could, for instance, be asked if all three functions have a maximum for the same parameter values. From the structure of the equation this can not be concluded, and, indeed, the question must be answered negatively because it can be shown that in general the maxima do not coincide. However, under assumptions that are usually fulfilled at least approximately, we can show that any global maximum of $\tilde{\mathcal{F}}(q,\Theta)$ is an approximate global maximum of $L_1(\Theta)$ and of $L_0(\Theta)$. A necessary condition for an approximate global maximum of $L(\Theta)$ (the likelihood of the original model) is thus a global likelihood maximum of $L_1(\Theta)$ (or of $L_0(\Theta)$). The technical derivation of this observation is given in Appendix A.1.

Note that, intuitively, it makes sense that the maximization of the likelihood $L_1(\Theta)$ results in parameters that can approximately maximize $L(\Theta)$. To see this consider the example of Figure 1. If the truncated generative model with $c=1$ is optimized on the truncated data class $c=1$, the displayed generative fields W are learned. These parameter values can be expected to also result in close to maximum likelihood values for the generative model with $c=0$ on data class $c=0$, and also correspond to approximately optimal likelihood values of the original generative model on the original data. The approximation improves with increasingly many data points. For this example, all parameters to approximately maximize $L(\Theta)$ can be recovered based on the necessary condition of maximizing $L_1(\Theta)$. The example of Figure 1 also demonstrates that the first variational step already significantly reduces computational costs. The truncated model with $c=1$ only requires the evaluation of 56 states per data point instead of $2^{10}=1024$ evaluations for the original model.

3.2 Second Variational Approximation: Preselection

Starting point for the second variational approximation will be the likelihood $L_1(\Theta)$ of the truncated generative model $c = 1$. We have seen in the previous section (and Appendix A.1) that a global maximum in $L_1(\Theta)$ is a necessary condition for an approximate global maximum of the likelihood $L(\Theta)$ of the original model. To find the maximum of $L_1(\Theta)$ we optimize the lower bound $\mathcal{F}_1(q,\Theta)$ given by:

$$\mathcal{F}_1(q,\Theta) \;=\; \overbrace{\sum_{n\in\mathcal{M}}\sum_{\vec{s}\in\mathcal{K}} q^{(n)}(\vec{s};\Theta^{\text{old}})\,\log\left(p(\vec{y}^{\,(n)}\,|\,\vec{s},\Theta)\,\frac{1}{\tilde{\kappa}}\,p(\vec{s}\,|\,\Theta)\right)}^{Q_1(q,\Theta)} + H(q)\,, \tag{17}$$

with $\sum_{\vec{s}\in\mathcal{K}} q^{(n)}(\vec{s};\Theta^{\text{old}}) = 1$. $\mathcal{F}_1(q,\Theta)$ is derived by a variational approximation, this time w.r.t. the hidden variables $\vec{s}$. The free-energy equals the likelihood $L_1(\Theta)$ after each E-step if the distributions $q^{(n)}(\vec{s};\Theta^{\text{old}})$ are given by:

$$q^{(n)}(\vec{s};\Theta^{\text{old}}) \;=\; p(\vec{s}\,|\,\vec{y}^{\,(n)},c=1,\Theta^{\text{old}}) \;=\; \frac{p(\vec{s}\,|\,\vec{y}^{\,(n)},\Theta^{\text{old}})}{\sum_{\vec{s}'\in\mathcal{K}} p(\vec{s}\,'\,|\,\vec{y}^{\,(n)},\Theta^{\text{old}})}\delta(\vec{s}\in\mathcal{K}). \tag{18}$$

M-step rules can be derived by setting the derivatives of $\mathcal{F}_1(q,\Theta)$ w.r.t. all parameters to zero. As the entropy term in (17) is independent of Θ if q is held fixed, we obtain

$$\frac{d}{d\Theta}\mathcal{F}_1(q,\Theta) = \frac{d}{d\Theta}Q_1(q,\Theta) = 0 \tag{19}$$

as necessary condition. The derivative $\frac{d}{d\Theta}$ hereby stands for derivatives w.r.t. all the individual parameters.

Based on condition (19) we can now introduce candidate preselection as a variational approximation. As described in Section 2, preselection amounts to selecting, for a given $\vec{y}^{\,(n)}$, a subset $\mathcal{K}_n$ of the state space. Section 2 gives an example of how to define the set $\mathcal{K}$ and how to construct subsets $\mathcal{K}_n$ using selection functions. Figure 2 shows a concrete example of how a set $\mathcal{K}_n$ is constructed using selection function values $\mathcal{S}_h(\vec{y}^{\,(n)})$. In Section 4 and Appendix B different instances of selection functions can be found. More generally, we here require from the sets $\mathcal{K}_n$ that for all data points generated by $\vec{s}\in\mathcal{K}$, they finally contain most of the posterior mass in $\mathcal{K}$. If this applies, we obtain an approximation to the posterior $q^{(n)}$ in (18) given by:

$$\tilde{q}^{(n)}(\vec{s};\Theta^{\text{old}}) \;=\; \frac{p(\vec{s}\,|\,\vec{y}^{\,(n)},\Theta^{\text{old}})}{\sum_{\vec{s}'\in\mathcal{K}_n} p(\vec{s}\,'\,|\,\vec{y}^{\,(n)},\Theta^{\text{old}})}\;\delta(\vec{s}\in\mathcal{K}_n) \;=\; \frac{p(\vec{s},\vec{y}^{\,(n)}\,|\,\Theta^{\text{old}})}{\sum_{\vec{s}'\in\mathcal{K}_n} p(\vec{s}\,',\vec{y}^{\,(n)}\,|\,\Theta^{\text{old}})}\;\delta(\vec{s}\in\mathcal{K}_n). \tag{20}$$

Note that $\tilde{q}^{(n)}(\vec{s};\Theta^{\text{old}})$ sums to one in $\mathcal{K}$ as $\mathcal{K}_n\subseteq\mathcal{K}$. It thus fulfils the condition on $q^{(n)}$ required for (17). If preselection finds, at least finally, appropriate sets $\mathcal{K}_n$, we obtain with (20) the necessary condition: $\frac{d}{d\Theta}Q_1(\tilde{q},\Theta)\approx\frac{d}{d\Theta}Q_1(q,\Theta)=0$. Parameter update rules derived from $\frac{d}{d\Theta}Q_1(\tilde{q},\Theta)=0$ can therefore be expected to (at least approximately) optimize the free-energy (15) and thus $L_1(\Theta)$. The update rules derived will contain expectation values (the sufficient statistics) of the form $\langle g(\vec{s})\rangle_{\tilde{q}^{(n)}}$. If we use (20) for these expectations we obtain:

$$\langle g(\vec{s})\rangle_{\tilde{q}^{(n)}} \;=\; \sum_{\vec{s}}\tilde{q}^{(n)}(\vec{s};\Theta^{\text{old}})\,g(\vec{s}) \;=\; \frac{\sum\limits_{\vec{s}\in\mathcal{K}_n} p(\vec{s},\vec{y}^{\,(n)}\,|\,\Theta^{\text{old}})\;g(\vec{s})}{\sum\limits_{\vec{s}'\in\mathcal{K}_n} p(\vec{s}\,',\vec{y}^{\,(n)}\,|\,\Theta^{\text{old}})}\,,$$

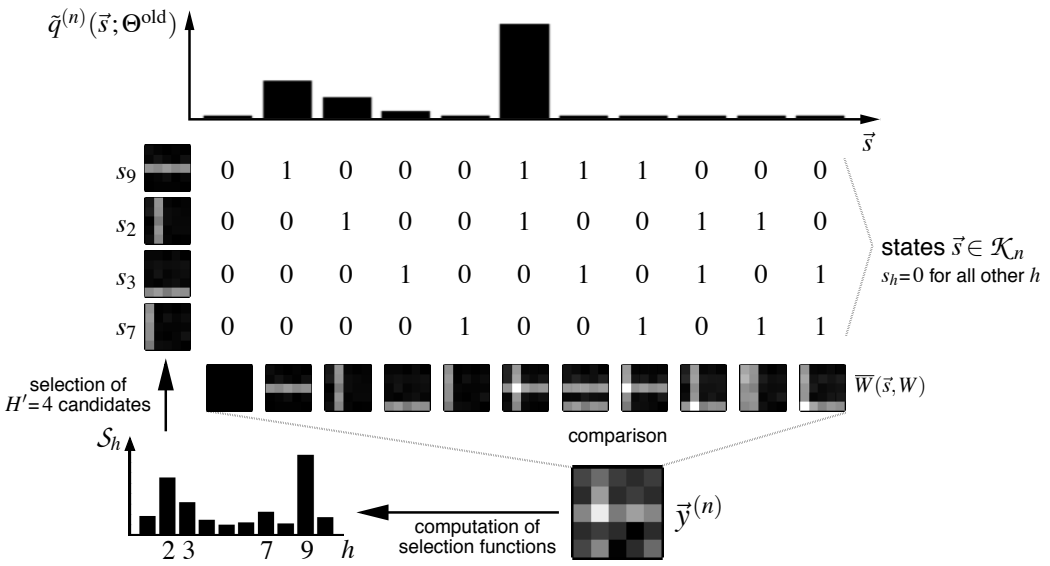


Figure 2: Second variational approximation: preselection. The figure illustrates how the variational approximation $\tilde{q}^{(n)}(\vec{s};\Theta^{\text{old}})$ in Equation 20 is computed. The selection of hidden states $\vec{s} \in \mathcal{K}_n$ and the computation of $\tilde{q}^{(n)}(\vec{s};\Theta^{\text{old}})$ are shown for the example of Figure 1 close to the optimal parameters W. Given the data point $\vec{y}^{(n)}$ a selection function value $\mathcal{S}_h$ for each hidden variable h is computed. The H' largest values are selected ($H' = 4$ for this example). The approximation $\tilde{q}^{(n)}(\vec{s};\Theta^{\text{old}})$ is then computed based on the combinatorics of these H' candidates (with $\sum_j s_j \leq \gamma = 2$). For the displayed data point and parameters all values of $\tilde{q}^{(n)}(\vec{s};\Theta^{\text{old}})$ except for one lie close to zero. For visualization purposes these values have been increase in the figure.

that is, precisely expression (6) in Section 2. Expectation Truncation, introduced as an approximation to Equation 5, can thus been derived as a variational approximation. Importantly, this approximation is tractable if $\mathcal{K}_n$ is small. The computational gain of preselection compared to an approximation without preselection is reflected by the reduced size of $\mathcal{K}_n$ compared to $\mathcal{K}$ (see Figure 2 for an example and Appendix C for a detailed complexity analysis).

3.3 Summary and Numerical Controls

We have seen that the approximation procedure introduced in Section 2 can be derived as a variational EM approach. This approach consists of two variational approximation steps: First, an approximation that assigns the data points to two classes (Section 3.1). Second, a variational step that approximates the posterior (18) by an approximate posterior (20) defined through preselection (Section 3.2). Although the derivation of ET as a variational approach requires in parts rather technical steps, the final result is intuitive (see Figure 1 and Figure 2) and can be stated very compactly. Algorithm 2 summarizes all required steps of the approximation scheme.

Note that also with preselection, the ET approximation still requires a summation over $\mathcal{K}$, namely $\sum_{\vec{s}\in\mathcal{K}} p(\vec{s}|\,\Theta)$. This sum has to be computed to determine N^{cut}, $N^{\text{cut}} = N\sum_{\vec{s}\in\mathcal{K}} p(\vec{s}|\,\Theta)$,

Algorithm 2: Expectation Truncation

Preselection:	select a state space volume $\mathcal{K}_n$ for each data point $\vec{y}^{(n)}$
Data classification:	find a data set $\mathcal{M}$ that approximates $\mathcal{M}^{\text{opt}}$ in Equation 16
E-step:	compute $\tilde{q}^{(n)}(\vec{s};\Theta^{\text{old}}) = \dfrac{p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^{\text{old}})}{\sum_{\vec{s}\in\mathcal{K}_n} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta^{\text{old}})}$ for all $\vec{y}^{(n)}$ and $\vec{s}\in\mathcal{K}_n$
M-step:	find parameters Θ such that $\dfrac{d}{d\Theta}\sum_{n\in\mathcal{M}}\sum_{\vec{s}\in\mathcal{K}_n}\tilde{q}^{(n)}(\vec{s};\Theta^{\text{old}})\,\log\left(p(\vec{y}^{(n)}\,|\,\vec{s},\Theta)\,\dfrac{p(\vec{s}\,|\,\Theta)}{\sum_{\vec{s}'\in\mathcal{K}}p(\vec{s}'\,|\,\Theta)}\right)=0$

and it appears in the M-step equation (Algorithm 2). Because of symmetries in the usual priors for generative models, this sum can, however, be computed without summing over all $\vec{s}$ explicitly (an example is given in the next section). Even if symmetries can not be exploited, note that the sum over $\vec{s}\in\mathcal{K}$ has to be computed at most once per EM iteration and not once per data point and per EM iteration (as it is the case for the sums over $\mathcal{K}_n$).

Algorithm 2 summarizes ET as a variational approximation and represents a generalization of Algorithm 1. Algorithm 1 in Section 2 contains, for example, one specific way of how to select an appropriate set $\mathcal{K}$ and appropriate sets $\mathcal{K}_n$: we defined $\mathcal{K}$ based on sparseness and selected $\mathcal{K}_n$ using selection functions $\mathcal{S}_h$. In the variational derivation of ET we, however, only specified the properties required from $\mathcal{K}$ and $\mathcal{K}_n$. In general, $\mathcal{K}$ does not have to be defined based on a sparseness assumption and there are potentially alternative ways to define the sets $\mathcal{K}_n$. Importantly, the variational derivation of ET allows for a comparison with other variational approaches. We can thus observe that ET is qualitatively different from the standard variational approach. Concrete instances of variational EM usually approximate the exact posteriors by fully or partly factored distributions over the hidden variables (compare Jordan et al., 1999; Jaakkola, 2000; MacKay, 2003; Bishop, 2006). Such approximations become the more severe the stronger dependencies between the hidden variables in the posterior are. In the derivation of the ET approximation, no independence assumption for the posterior has been used. Strong dependencies are therefore not expected to negatively affect the approximation quality of ET. On the other hand, the ET approximations can get more severe if the approximate classification of data points becomes imprecise (first variational step), if the preselection step does not include the relevant candidates (second variational step), or if too few data points are used.

ET has in common with all variational EM approaches that there is no guarantee for the likelihood to finally increase to values close to the global optimum. This can in general be due to the approximations being too severe or due to many local optima in the likelihood landscape. Variational approximations therefore have to be verified numerically. In the following section, ET will be evaluated based on different generative models and different data sets. During likelihood maximization we will monitor different values relevant for the approximation. We will thus control if the likelihood is indeed increased during learning and, given ground-truth, if it approaches values close to the global likelihood maximum. Ground-truth data will also allow us to quantify how well the generating parameters are recovered by the derived algorithm. Furthermore, we will monitor values that give evidence about the quality of the specific approximations used by ET. We will thus

monitor the quality of the data point classification (first variational approximation) and the quality of the approximation by preselection (second variational approximation). For the first approximation we compare the obtained classification with ground-truth of all data points. For the second approximation, we will monitor the differences between the exact posterior $p(\vec{s}\,|\,\vec{y}^{(n)},\Theta)$ and the approximate posterior $\tilde{q}^{(n)}(\vec{s};\Theta)$ in (20). If we evaluate the differences between these distributions using the Kullback-Leibler divergence, we obtain:

$$D_{KL}(\tilde{q}^{(n)},p) = -\log(Q^{(n)}) \quad \text{with} \quad Q^{(n)} = \frac{\sum_{\vec{s}\in\mathcal{K}_n} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta)}{\sum_{\vec{s}'} p(\vec{s}',\vec{y}^{(n)}\,|\,\Theta)}\,. \tag{21}$$

The preselection approximation has a high quality if for the data points in $\mathcal{M}$ the values $D_{KL}(\tilde{q}^{(n)},p)$ are close to zero. For data points not in $\mathcal{M}$ the differences between the distributions should be large. For practical reasons, we have introduced the quality values $Q^{(n)}$ in (21). The values $Q^{(n)}$ measure the percentage of the posterior probability mass concentrated in $\mathcal{K}_n$. In terms of these values the approximation quality is high if $Q^{(n)}$ is close to one for data points in $\mathcal{M}$, and close to zero for data points not in $\mathcal{M}$.

4. Training Generative Models

The approximation scheme defined and discussed in the previous sections is so far independent of the specific choice of a generative model except for the assumption of binary hidden variables. To demonstrate the applicability of the method and to investigate its properties, in this section we will apply it to a number of different generative models. The investigated models are all multiple-cause models that require tractable approximations. The three major classes investigated here are non-negative matrix factorization (NMF; Section 4.1), maximal causes analysis (MCA; Section 4.2), and a sparse-coding-like model termed LinCA (Section 4.3). For all these models we will assume independent hidden variables distributed according to a Bernoulli prior:

$$p(\vec{s}\,|\,\pi) = \prod_{h=1}^{H} p(s_h\,|\,\pi), \quad p(s_h\,|\,\pi) = \pi^{s_h}\,(1-\pi)^{1-s_h}\,, \tag{22}$$

where $\pi \in [0,1]$ parameterizes the sparseness of the distribution. For binary hidden variables the Bernoulli prior represents the most straightforward choice (compare, e.g., Berkes et al., 2009; Lücke and Sahani, 2008). Given the prior (22) the expected number of data points generated by less or equal γ causes is given by:

$$N^{\leq\gamma} = N \sum_{\vec{s},\,|\vec{s}|\leq\gamma} p(\vec{s}\,|\,\pi) = N \sum_{\gamma'=0}^{\gamma} \left(\begin{array}{c} H \\ \gamma' \end{array}\right) \pi^{\gamma'}\,(1-\pi)^{H-\gamma'}\,. \tag{23}$$

$N^{\leq\gamma}$ is required to find the set of N^{cut} data points $\mathcal{M}$ considered for a parameter update, and Equation 23 shows that this number is tractably computable. For all generative models we will use ET as described in Section 2 and Algorithm 1. That is, we will use a set $\mathcal{K}$ that constrains the number of simultaneously active causes, and use selection functions $\mathcal{S}_h$ to obtain sets $\mathcal{K}_n$.

4.1 Non-negative Matrix Factorization (NMF)

The first generative model considered uses prior (22) and combines the generative fields $\vec{W}_h = (W_{1h},\ldots,W_{Dh})^T$ of all latents with $s_h = 1$ linearly. We use a Gaussian noise model such

that the probability of a data vector $\vec{y}$ given $\vec{s}$ is defined by:

$$p(\vec{y}\,|\,\vec{s},\Theta) = \prod_{d=1}^{D} p(y_d\,|\,\overline{W}_d(\vec{s},W),\sigma), \quad p(y_d\,|\,w,\sigma) = \mathcal{N}(y_d;w,\sigma^2) \tag{24}$$

$$\text{with} \quad \overline{W}_d(\vec{s},W) = \sum_h W_{dh} s_h\,, \tag{25}$$

with $W \in \mathbb{R}^{D\times H}$. $\mathcal{N}(y_d;w,\sigma^2)$ is a scalar Gaussian density function with mean w and variance σ^2. The introduction of $\overline{W}$ will turn out to be convenient for the analytical treatment below. Note that we otherwise could have written $p(\vec{y}\,|\,\vec{s},\Theta) = \mathcal{N}(\vec{y};W\vec{s},\sigma^2\mathbb{1})$.

We then use (24) to compute the update equation (the M-step) for the weight matrix W as described in Section 2. The update equation is gained from the necessary condition for an optimum of the free-energy in (2). The W-update is consequently a function of the sufficient statistics $\langle\vec{s}\rangle^T_{q^{(n)}}$ and $\left\langle\vec{s}\vec{s}^{\,T}\right\rangle_{q^{(n)}}$. Following Section 2, these expectation values are approximated by using (6) instead of the exact sufficient statistics (5), that is, we use a subset of cause combinations as selected by the truncation approach. Furthermore, we sum only over the N^{cut} data points in $\mathcal{M}$ as explained at the end of Section 2. The update equation is then given by:

$$W \;=\; \left(\sum_{n\in\mathcal{M}} \vec{y}^{\,(n)}\,\langle\vec{s}\rangle^T_{q^{(n)}}\right)\left(\sum_{n'\in\mathcal{M}} \left\langle\vec{s}\vec{s}^{\,T}\right\rangle_{q^{(n')}}\right)^{-1}. \tag{26}$$

Note that the equation can consistently be gained from the necessary condition for a free-energy optimum in Equation 19 (see M-step of Algorithm 2).

Equations 24 to 26 are valid irrespective of the sign of the entries of the generative fields and the input data. Generative models corresponding to the class of non-negative Matrix Factorization (NMF) methods are based on a linear combination of generative fields but rely on non-negative data points and generative fields. In Equation 26, non-negativity can be ensured for the generative fields by clamping small appearing weights at zero.

A more direct way to ensure non-negativity for the parameters of the model defined by (22) and (24) is to rely on convergence proofs similar to those used for classical NMF (Lee and Seung, 2001), that is, to ensure non-negativity by deriving a multiplicative update rule for the generative fields. For the EM algorithm used as a basis for ET, it can be shown that the parameter-free update rule

$$\vec{W}_h \leftarrow \vec{W}_h \odot \frac{\langle\vec{y}\,s_h\rangle_{\text{ET}}}{\sum_{h'}\vec{W}_{h'}\,\langle s_{h'}\,s_h\rangle_{\text{ET}}} \tag{27}$$

provides monotonic convergence towards the M-step solution of the generative fields, where we have introduced the ET-averaging

$$\langle f(\vec{y},s_h)\rangle_{\text{ET}} := \sum_{n\in\mathcal{M}}\sum_{\vec{s}\in\mathcal{K}_n} p(\vec{s}|\vec{y}^{\,(n)},W)\,f(\vec{y}^{\,(n)},s_h) = \sum_{n\in\mathcal{M}}\left\langle f(\vec{y}^{\,(n)},s_h)\right\rangle_{q^{(n)}}.$$

Equation 27 corresponds to a partial M-step of ET-NMF ('Expectation-Truncation-NMF'). In simulations, it is therefore applied iteratively with 20 partial M-steps after each E-step.

For the introduced generative model, Equation 27 converges towards the M-step solutions of the EM algorithm under non-negativity constraints. The update rule can be understood as a diagonally rescaled gradient descent derived from the EM algorithm, with a rescaling factor that is "optimally

chosen to ensure convergence" (Lee and Seung, 2001). A thorough derivation of the update Equation 27 and its relation to the classical NMF algorithm known from the literature can be found in Appendix D.

The E-step of ET-NMF is based on the truncated expectation values (6) to calculate the averaged quantities in the W update equations according to

$$\langle \vec{y}\, s_h \rangle_{\mathrm{ET}} = \sum_{n\in\mathcal{M}} \vec{y}^{(n)} \langle s_h \rangle_{q^{(n)}} \qquad \text{and} \qquad \langle s_{h'}\, s_h \rangle_{\mathrm{ET}} = \sum_{n\in\mathcal{M}} \langle s_{h'}\, s_h \rangle_{q^{(n)}}$$

so that the sufficient statistics of the ET-NMF model that have to be computed for the M-step will be given by the first and second order moments $\langle s_h \rangle_{q^{(n)}}$ and $\langle s_{h'}\, s_h \rangle_{q^{(n)}}$ of the (approximate) posterior.

For our generatively formulated version of NMF with M-step equations (26) or (27) we can now apply ET as described in Section 2. We ran the ET-NMF algorithm with both M-step versions (26) and (27) and observed, at least for the data used, a qualitatively and quantitatively comparable behavior. Note that the probability density $p(\vec{y}\,|\,\Theta)$ of the model is invariant under the exchange of any two generative fields (or basis functions), $\vec{W}_h \rightarrow \vec{W}_{h'}$, because of symmetric priors. By the definition of the truncated generative models (Section 3), it can instantly be seen that their probability densities $p(\vec{y}\,|\,c=1,\Theta)$ and $p(\vec{y}\,|\,c=0,\Theta)$ are also invariant under these transformations (the same will apply for the other models considered).

As indicated, the sufficient statistics for ET-NMF are given by the first and second order moments, $\langle s_h \rangle_{q^{(n)}}$ and $\langle s_{h'}\, s_h \rangle_{q^{(n)}}$, of the exact posterior; to find approximations to these intractable expectation values we first have to find appropriate selection functions $\mathcal{S}_h$. A natural starting point for finding such functions is to consider the joint probability $p(s_h=1,\vec{y}^{(n)}\,|\,\Theta) = \sum_{\vec{s}\,(s_h=1)} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta)$. If we knew that the joint probability was small for a given h, we would know that the sum over all $\vec{s}$ in (6) which contains $\vec{s}$ with $s_h = 1$ is small as well. Furthermore, note that for a given data point the joint represents the part of $p(s_h=1\,|\,\vec{y}^{(n)},\Theta)$ which depends on h, $p(s_h=1\,|\,\vec{y}^{(n)},\Theta) = \frac{p(s_h=1,\vec{y}^{(n)}\,|\,\Theta)}{p(\vec{y}^{(n)}\,|\,\Theta)}$. $p(s_h=1\,|\,\vec{y}^{(n)},\Theta)$, on the other hand, directly reports the probability of unit h to have contributed to the data point. It could thus be regarded as the optimal selection function. Unfortunately, neither $p(s_h=1,\vec{y}^{(n)}\,|\,\Theta)$ nor $p(s_h=1\,|\,\vec{y}^{(n)},\Theta)$ are computationally tractable and, thus, neither function fulfils one of the properties demanded from a selection function. Therefore, we will compute an upper bound of $p(s_h=1,\vec{y}^{(n)}\,|\,\Theta)$ which is tractable and still serves well in selecting a subset of hidden units that can explain a given $\vec{y}^{(n)}$. Let us for this purpose consider the data point dependent weight matrix defined by: $W^{\mathrm{ub}}_{dh} := W^{\mathrm{ub}}_{dh}(\vec{y}^{(n)},W) = \max\{y^{(n)}_d, W_{dh}\}$ and $\vec{W}^{\mathrm{ub}}_h := \left(W^{\mathrm{ub}}_{1h},\ldots,W^{\mathrm{ub}}_{Dh}\right)^T$, where 'ub' refers to 'upper bound'. This formal definition of W^{ub} allows for a compact notation of an upper bound of $p(s_h=1,\vec{y}^{(n)}\,|\,\Theta)$. Because of the non-negativity of the entries in W and the mono-modality of the Gaussian distribution w.r.t. the mean, we can show (see Appendix B for details):

$$p(s_h=1,\vec{y}^{(n)}\,|\,\Theta) = \sum_{\vec{s}\,(s_h=1)} p(\vec{y}^{(n)}\,|\,\overline{W}_d(\vec{s},W),\sigma)\, p(\vec{s}\,|\,\pi) \;\leq\; \pi\, p(\vec{y}^{(n)}\,|\,\vec{W}^{\mathrm{ub}}_h,\sigma) =: \mathcal{S}_h(\vec{y}^{(n)})\,. \tag{28}$$

The upper bound $\mathcal{S}_h$ is tractable (also compare Appendix C) because we have removed the summation over the $\vec{s}$. The price we pay is that the selection function $\mathcal{S}_h$ can be a relatively coarse estimate in some cases. Importantly, however, we know that if $\mathcal{S}_h$ is sufficiently small, then the contribution of all joint probabilities $p(\vec{s},\vec{y}^{(n)}\,|\,\Theta)$ with $s_h = 1$ can be neglected. The E-step given by the approximation of the sufficient statistics in Section 2 with (28) as selection function together with

the M-step in (27) or (26) represents the learning algorithm for the NMF generative model defined by (22), (24) and (25) with non-negativity constraint.

In addition we add after each M-step a small parameter noise to the basis vectors W (iid Gaussian, standard deviation 0.05) and we use a standard relaxation scheme in order to avoid local optima of the potentially multi-modal likelihood landscape. Annealing (see, e.g., Ueda and Nakano, 1998; Sahani, 1999) amounts to the replacements: $(1/\sigma^2) \to (\beta/\sigma^2)$, $\pi \to \pi^\beta$ and $(1-\pi) \to (1-\pi)^\beta$. The constant β is an inverse 'temperature', $\beta = 1/T$, where T is decreased from a high value T^{init} to a value T^{final} equal or close to one.

4.1.1 EXPERIMENTS - ARTIFICIAL DATA

Let us consider data as displayed in Figure 3A. That is, we consider hidden causes in the form of horizontal and vertical bars (five pixels each) on a 5×5 grid. Each bar appears with probability $\frac{2}{10}$ such that there are on average two bars per data point. We use $N = 500$ data points. The grey-value of a bar is taken to be 10, background pixels are zero. The bars superimpose linearly (pixel values in regions of overlap are 20) and are subject to Gaussian noise with standard deviation $\sigma = 2.0$. Data as in Figure 3A are well-suited to study the functioning of the approximation scheme because we know the underlying generating process and have ground-truth for each data point. We will later use this knowledge to illustrate the influence of each data point on the update of the model parameters. For this reason data points such as displayed in Figure 3A or versions without noise are frequently used in the recent literature (see, e.g., Hinton et al., 1995; Hoyer, 2002; Spratling, 2006).

The model which is applied to the data uses $H = 10$ hidden units and $D = 25$ observed units. The entries of W are initialized by drawing iid from a Gaussian distribution with a mean of 4 and a standard deviation of $\frac{4}{3}$. The standard deviation of the generative model is set to $\sigma = 2.0$ and the value of π is set to $\frac{2}{10}$. Small deviations from these values did not have significantly negative effects on the algorithms performance in extracting data components. Figure 4B shows the cooling schedule for annealing. We linearly decrease T from $T^{\text{init}} = 13$ to $T^{\text{final}} = 1$ during 100 EM-iterations (including ten initial iterations at T^{init} and twenty final iterations at T^{final}). Figure 3B shows a typical time-course of the parameters W if trained as described above. As approximation parameters we used $H' = 5$ and $\gamma = 3$. Although approximate EM schemes do in general not guarantee the increase of the data likelihood, we find that the learning algorithm increases the likelihood to values close to the one for the generating parameters (dashed horizontal line in Figure 4A). This behavior is reflected by the convergence of the model parameters to values close to the generating ones (compare Figure 3). In most trials the parameters converged to approximately optimal values relatively early but in some trials they converged relatively late during learning (compare likelihood values in Figure 4A). We ran 50 trials with 50 different sets of data points generated as described above. The algorithm extracted all bars in all of the trials (see Appendix E for measurement details). To quantify the quality of parameter reconstruction we computed, for each trial, the mean absolute error (MAE) between the obtained generative fields, $\vec{W}_h$, and the corresponding generating causes: $\text{MAE} = \frac{1}{HD}\sum_{hd}|W_{dh} - W^{\text{gen}}_{dh'}|$ where $W^{\text{gen}}_{dh'}$ denotes the cause represented by $\vec{W}_h$ (compare Appendix E). For all trials the MAE was smaller than 0.24 with a mean of 0.20 (note that pixels of bars were set to 10.0 and background to zero).

In a second series of measurements we used the model with the same parameters and the same data as above except that the generating noise variance was set to zero. In 50 trials on this non-noisy data the algorithm extracted all bars in 46 of the trials (92% reliability) and extracted nine of the ten

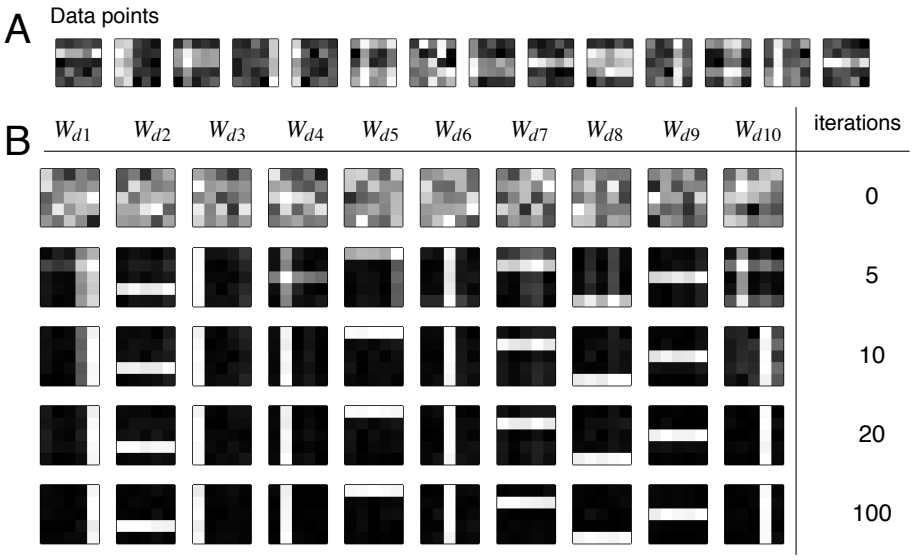


Figure 3: **A** 14 data points of the linear bars test with Gaussian noise. **B** Time course of the generative fields W of the NMF-like generative model if Expectation Truncation is used for learning.

single bars in the four remaining cases. All successful trials had a MAE of below 0.20 with a mean MAE of 0.05. Parameter recovery was more precise than in the previous trial series because of the non-noisy data. A higher initial temperature or a longer cooling increased reliability to values close to 100% for this data. E.g., when we used $T^{\text{init}} = 15$ and stretched the cooling schedule in Figure 4B to 200 iterations, the algorithm found all bars in all of 50 trials. Likewise, increasing the number of data points increased reliability ($> 94\%$ reliability for $N > 1000$ data points with $T = 13$ and 100 iterations cooling).

High reliability (i.e., a high probability to extract all causes) in this linear bars test has also been observed for other learning algorithms (see, e.g., algorithms investigated in Spratling, 2006). Note, however, that the standard bars benchmark test (Földiák, 1990) uses non-linearly overlapping bars (we will come to the standard version of the bars test in the next section). For the presented NMF algorithms we have (as is usual in the literature) only inferred the parameters W. In our generative setting it is in principle also possible to learn the model parameters σ and π (compare Lücke and Sahani, 2008). However, for comparison with other approaches and for brevity, we focused on W.

To investigate the quality of the ET approximation more directly, the values $Q^{(n)}$ (Equation 21) were computed for 40 data points during learning. Figure 4C shows time courses of $Q^{(n)}$ during a trial on the noisy linear bars test using the parameters given above. The data points were randomly selected but it was made sure that twenty of them were generated by less or equal γ causes (bright green lines) and the other twenty by more than γ causes (dark red lines). As can be observed, the approximation quality of the twenty data points generated by $\leq \gamma$ causes quickly increases whereas the approximation for the other twenty data points approaches zero. From the used approximation this could have been expected as we have restricted the summation in (6) to hidden states with less

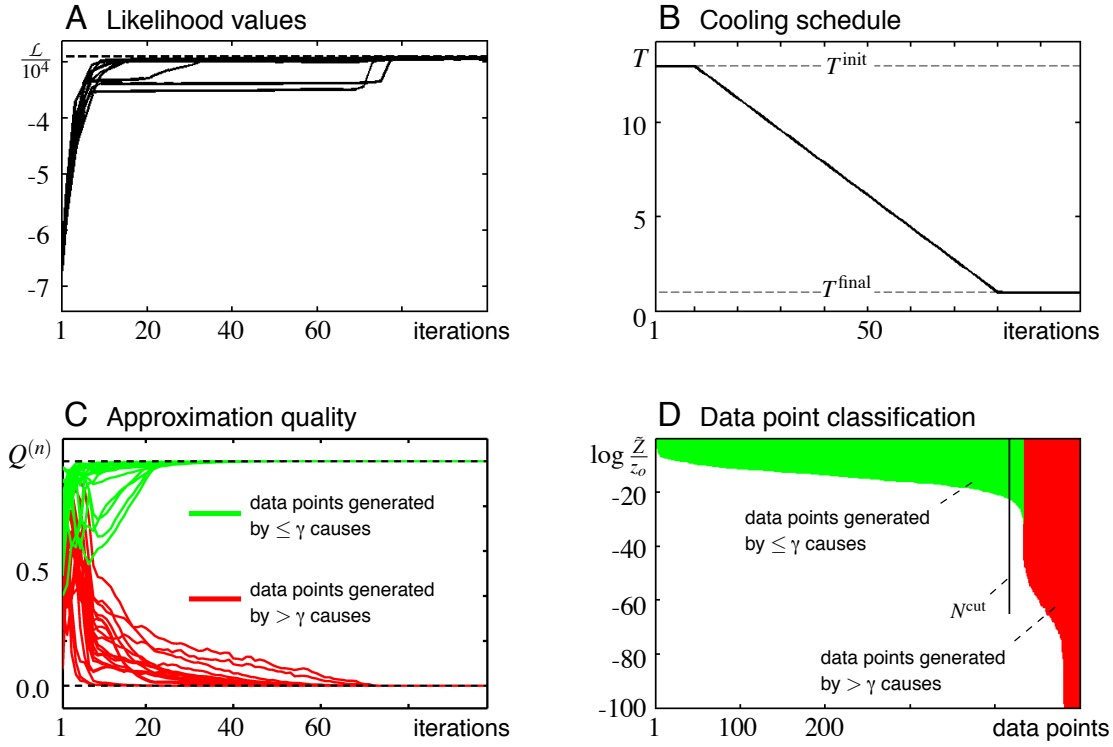


Figure 4: **A** Data likelihood during ten trials using the same set of $N = 500$ data points but different random initializations. The likelihood value that results from using the generating parameters is marked by the dashed horizontal line. **B** Cooling schedule during learning. **C** Approximation qualities $Q^{(n)}$ of 40 data points during learning. Twenty of the data points were generated by $\leq \gamma$ causes (bright green lines) and the other twenty by more than γ causes (dark red lines). **D** Sorted values of the sums $\tilde{Z} = \sum_{\vec{s} \in \mathcal{K}_n} p(\vec{s}, \vec{y}^{(n)} \,|\, \Theta)$ at the end of learning. Bright green bars were used to mark the data points generated by $\leq \gamma$ causes, dark red bars to mark data points generated by more than γ causes. The N^{cut} data points left of the black vertical line were used in the final M-steps.

or equal $\gamma = 3$ active causes. For other trials, the values $Q^{(n)}$ show the same qualitative behavior. However, the exact time-courses can differ quantitatively from trial to trial.

Note that the poorly approximated data points do finally not negatively affect the parameter updates because they are not taken into account for the M-step. This is illustrated in Figure 4D which shows the logarithms of the sum $\tilde{Z} = \sum_{\vec{s} \in \mathcal{K}_n} p(\vec{s}, \vec{y}^{(n)} \,|\, \Theta)$ at the end of learning and for each of the $N = 500$ data points used (divided by a common factor z_o). Bright green bars display the values of all data points generated by less or equal γ causes, dark red bars display the values of all other data points. The data points are ordered descendingly. According to the approximation used (see Section 2 resp. Equation 23), we only consider the N^{cut} data points left of the black bar, that is, we finally only use data points with quality values $Q^{(n)}$ close to one. The relation of the quality values to the KL-divergence in (21) directly shows that the ET approximation for these data points is virtually optimal in this case.

4.1.2 EXPERIMENTS - MORE REALISTIC DATA

As a second example, we applied the learning algorithm to gray-valued images of the postal digits database MNIST (http://yann.lecun.com/exdb/mnist/). This data is frequently used in the NMF literature, which makes it well suited as a means for comparison of our algorithm to standard NMF approaches. Note that we do not have ground-truth about the hidden components in this case.

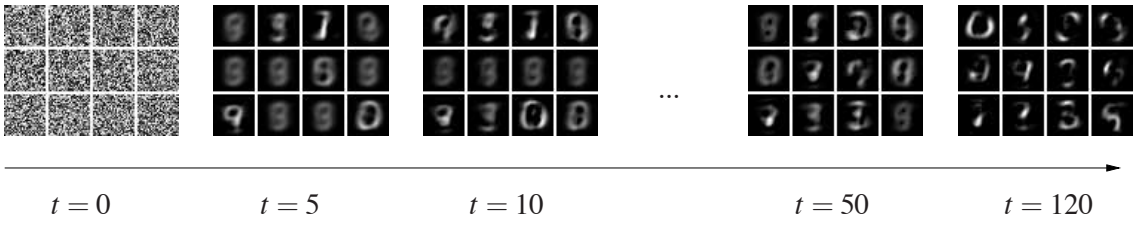


Figure 5: Resulting basis vectors $\vec{W}_h$ for the MNIST database. The probabilistic version of NMF trained with ET converges to a parts-based decomposition.

Figure 5 shows the application of the algorithm using $H = 12$ hidden variables and approximation parameters for ET set to $H' = 10$ and $\gamma = 5$. The prior parameter was set to $\pi = 0.3$ such that three to four of the 12 latent variables do explain a data point on average. The noise parameter σ in (24) was set to $\sigma = 0.73$ after screening through values between zero and one. The dimensionality of each data point is $D = 28 \times 28$ and we used a subset of $N = 1000$ data points, some of which are shown in Figure 6A. We linearly decreased the temperature as in Figure 4B but used a slightly longer learning time (120 iterations) to provide more time for convergence. In Figure 5 the time course of W displayed as basis vector sets is shown. As can be observed, the parameters W converge to basis vectors that represent digit parts.

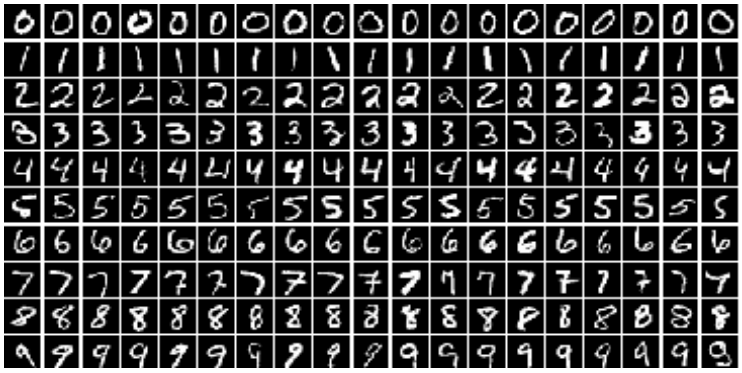
To assess the quality of the basis vectors and for comparison with standard NMF, we show average reconstructions of probabilistic NMF and standard NMF in Figure 6. In Figure 6B it can be observed that already for the small subset of 12 basis vectors in Figure 5 the reconstructions match the inputs in Figure 6A relatively well. Despite the constraint to binary hidden variables in our generative version of NMF, the resulting reconstructions are very similar to those of standard NMF as shown in Figure 6C. For these data, the overall average reconstruction error, $\frac{1}{ND}\sum_n ||\vec{y}^{(n)} - \sum_h \vec{W}_h \langle s_h \rangle_{q^{(n)}}||^2$, of the generative version is less than 5% larger than the reconstruction error of standard NMF.

4.2 Maximal Causes Analysis (MCA)

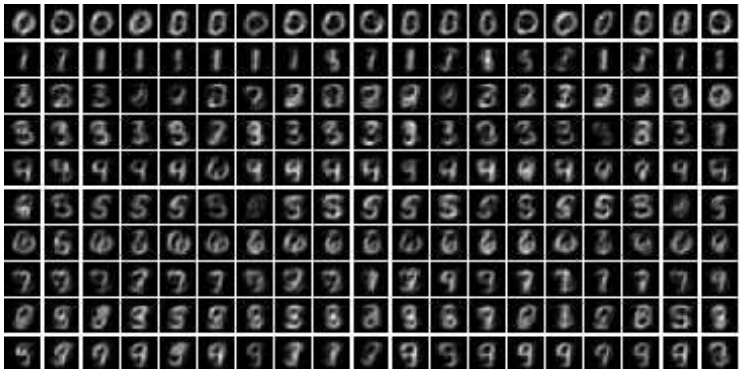
The second generative model we consider was suggested to extract the hidden causes from data whose components combine non-linearly. It uses a maximum rule in the place where NMF, sparse coding (Olshausen and Field, 1996), independent component analysis (ICA; Comon, 1994) and many other methods assume a linear superposition of hidden components:

$$p(\vec{y}\,|\,\vec{s},W) = \prod_{d=1}^{D} p(y_d\,|\,\overline{W}_d(\vec{s},W),\sigma), \quad p(y_d\,|\,w) = \mathcal{N}(y_d; w, \sigma^2) \tag{29}$$
$$\text{with} \quad \overline{W}_d(\vec{s},W) = \max_h\{s_h W_{dh}\},$$

A Digit data used for testing the ET-version of NMF



B ET-NMF average reconstruction



C Standard NMF reconstruction

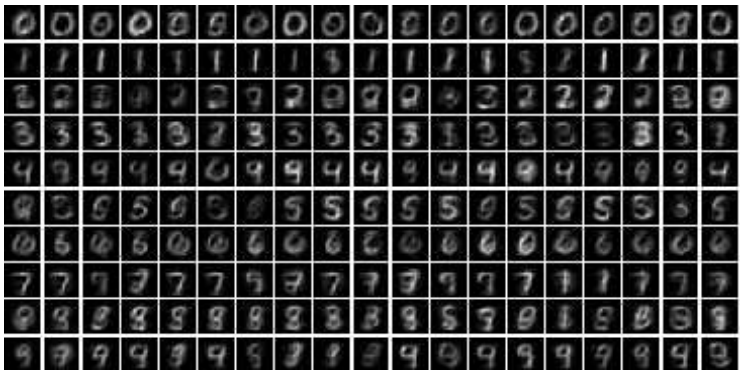


Figure 6: Reconstruction of data points representing hand-written digits. **A** Subset of the $N = 1000$ data points used for the application of probabilistic and standard NMF to the MNIST data base. **B** Average reconstruction of the digit data in **A** on the basis of the basis vectors in Figure 5. **C** Reconstruction of the digit data in **A** using standard NMF with the same number of basis vectors.

where we have used a Gaussian noise model with w as a place-hoder for $\overline{W}_d(\vec{s},W)$ (note the difference to Poisson noise used by Lücke and Sahani, 2008). For the activities of the binary hidden variables s_h we use the same prior as for the NMF model in Section 4.1 (see Equation 22). As in the previous model, the weight matrix $W \in \mathbb{R}^{D\times H}$ parameterizes the influence of the hidden causes on the distribution of $\vec{y}$. The function $\overline{W}_d(\vec{s},W)$ in (29) gives the *effective weight* on y_d, resulting from a particular instance of the state vector $\vec{s}$. An update rule for the weight matrix W of this model was derived in Lücke and Sahani (2008) and is given by:

$$W_{dh} = \frac{\sum\limits_{n\in\mathcal{M}} \left\langle \mathcal{A}^{\rho}_{dh}(\vec{s},W)\right\rangle_{q^{(n)}} y^{(n)}_d}{\sum\limits_{n\in\mathcal{M}} \left\langle \mathcal{A}^{\rho}_{dh}(\vec{s},W)\right\rangle_{q^{(n)}}}, \quad \text{where} \qquad \begin{aligned} \mathcal{A}^{\rho}_{dh}(\vec{s},W) &= \left(\frac{\partial}{\partial W_{dh}}\overline{W}^{\rho}_d(\vec{s},W)\right) && (30)\\ \overline{W}^{\rho}_d(\vec{s},W) &= \left(\sum_{h=1}^{H}(s_h W_{dh})^{\rho}\right)^{\frac{1}{\rho}}. && (31)\end{aligned}$$

The parameter ρ controls the nonlinearity and is increased to large values during learning. Again, the entries in W are non-negative. To derive a selection function we can therefore apply the same arguments as for NMF and thus arrive at the very same functions $\mathcal{S}_h$ as given in Equation 28. The selection function (28), M-step equations (30) and (31), and the E-step approximation described in Section 2 represent a full learning algorithm for the extraction of non-linearly combining components, which will be referred to as MCA_{ET}.

4.2.1 EXPERIMENTS - ARTIFICIAL DATA

To study the properties of MCA_{ET} let us first apply it to data with ground-truth. A well-suited type of data for the algorithm is the so-called bars test introduced by Földiák (1990). The bars test has become a standard benchmark for component extraction algorithms (see, e.g., Saund, 1995; Dayan and Zemel, 1995; Hochreiter and Schmidhuber, 1999; Charles et al., 2002; Lücke and von der Malsburg, 2004; Spratling, 2006; Butko and Triesch, 2007; Lücke and Sahani, 2008) and thus allows for quantitative comparison with other systems. To generate data according to the bars test we use the same parameter settings as for the artificial data in Figure 3A, that is, $D = 5\times 5$, bars pixel value 10 and other pixels zero, Gaussian generating noise with standard deviation 2.0, and the probability for each bar to occur is $\frac{2}{10}$. In contrast to the data used in the experiment of Figure 3, however, the standard form of the bars test uses a non-linear superposition of the causes (overlapping bar regions have pixel values 10 instead of 20 for NMF). Figure 7A shows a random selection of 12 of the $N = 500$ data points used.

We apply MCA_{ET} to the data using the same model parameters and the same approximation parameters ($H' = 5$ and $\gamma = 3$) as for the linear bars test in Section 4.1.1. Annealing for MCA_{ET} amounts to the same replacements as for NMF: $(1/\sigma^2) \to (\beta/\sigma^2)$, $\pi \to \pi^{\beta}$ and $(1-\pi) \to (1-\pi)^{\beta}$. Additionally, ρ in (30) and (31) is increased from a relatively small value at T^{init} to a large value at T^{final} by making ρ temperature dependent: $\rho = \frac{1}{1-\beta} = \frac{T}{T-1}$. As cooling schedule we use the same one as in Section 4.1.1 (see Figure 4B) but with $T^{\text{final}} = 1.05$ to avoid a singularity for ρ. For MCA we found it beneficial to add to the set $\mathcal{K}_n$ (Equation 7), the set of all vectors with just one non-zero entry: $\overline{\mathcal{K}}_n = \mathcal{K}_n \cup \{\vec{s} \mid \sum_i s_i = 1\}$. Making $\mathcal{K}_n$ larger can in general only increase the accuracy of the approximation. At the same time, using $\overline{\mathcal{K}}_n$ instead of $\mathcal{K}_n$ does not change the scaling behavior with H of the algorithm (see Appendix C for a discussion).

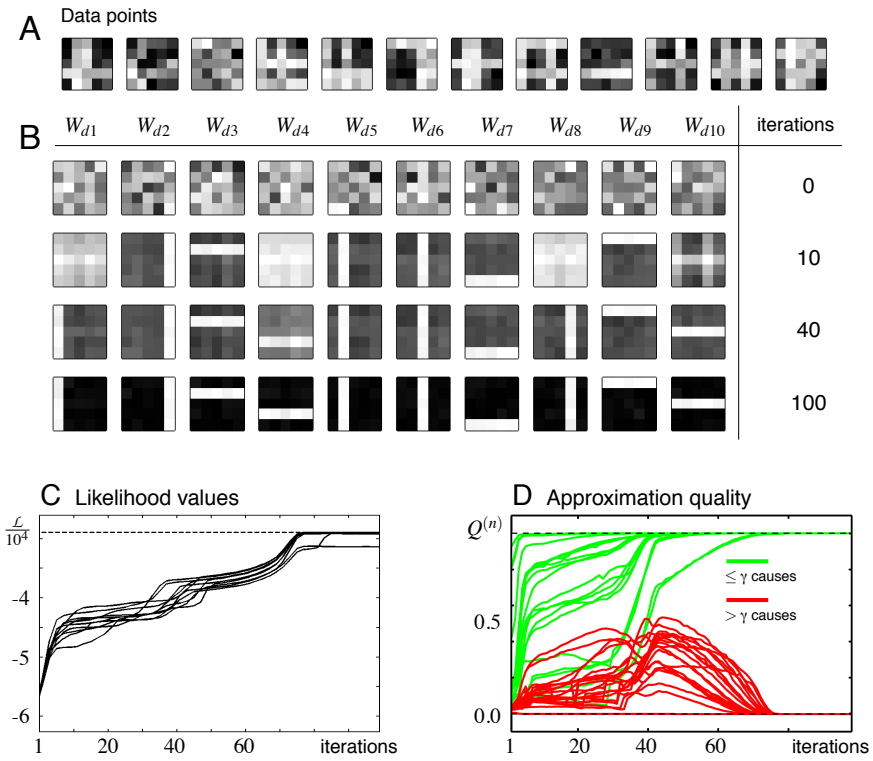


Figure 7: **A** Example data of a bars test with $D = 5 \times 5$ and additive Gaussian noise. **B** Time course of W for the MCA model trained with ET. **C** Likelihood values for ten trials with the same set of training patterns and different initial conditions for each trial. **D** Time course of the quality values of 40 data points during one trial. Values are plotted for twenty randomly selected data points generated by less or equal γ causes (bright green lines) and twenty randomly selected data points generated by greater than γ causes (dark red lines).

Figure 7B shows a typical time course of W during learning. Figure 7C shows time courses of the data likelihood for ten different runs using the same data set. The behavior of the likelihood values results from the specific form of annealing which includes the annealed nonlinearity in Equation 31. In Figure 7D typical time courses for the quality values $Q^{(n)}$ (Equation 21) for 40 data points are shown. For the 20 data points which were generated by $\leq \gamma$ causes (bright green lines) the quality values increased to one. For the data points generated by $> \gamma$ causes (dark red lines) the quality values finally decreased to zero. As for NMF, only the data points which were well approximated were finally taken into account for learning.

To probe the reliability of MCA_{ET}, we ran 50 trials of the bars test with the bars test parameters as given above. In each trial we used a new set of $N = 500$ data points. In 46 of the 50 trials MCA_{ET} extracted all bars (92% reliability), and in four of the trials 9 of 10 bars were extracted. Reconstruction of the generating parameters was high with a maximal MAE of 0.35 and a mean

MAE of 0.29 (bars had value 10). We observed that the convergence to local optima in 8% of the trials was mainly due to effects of finite sample size. When we ran 100 trials using the same parameters but $N = 2000$ data points instead of $N = 500$, the algorithm extracted all bars in all trials.

For comparison with other methods, we ran additional trials using the same parameters for bars generation but with Gaussian noise for data pixels set to variance zero. This version of the bars test is presumably the one most commonly used in the literature (e.g., Saund, 1995; Dayan and Zemel, 1995; Hochreiter and Schmidhuber, 1999; Lücke and Sahani, 2008). In 41 of the 50 trials MCA_{ET} extracted all bars (82% reliability) and found 9 of 10 bars in the other nine trials. Again, reconstruction of the generating parameters in the successful trials was high with a maximal MAE of 0.14 and a mean MAE of 0.05. Also for the non-noisy bars test, reliability of the algorithm increased when we increased the sample size. For instance, we obtained reliabilities of more than 90% for N larger than about 2000. With $N = 10\,000$ and slower cooling (same cooling schedule as in Figure 4B but stretched to 200 iterations), the algorithm found all bars in all of the 50 trials. Achieving close to 100% reliability is thus more difficult for non-noisy bars.[1]

In earlier work, generative modelling approaches to the bars test merely achieved relatively low reliability values. For instance, the model of Saund (1995) achieved 27% reliability, and the model of Dayan and Zemel (1995) (although trained without overlapping bars) achieved 69%. Approaches such as PCA or ICA that assume linear superposition have been reported to fail in this task (see Hochreiter and Schmidhuber, 1999). Other objective function approaches and different types of neural network approaches (e.g., Charles et al., 2002; Lücke and von der Malsburg, 2004; Spratling, 2006, and references therein) have been more successful in terms of reliability. They do, however, often use hidden assumptions and constraints which make an objective comparison difficult (see, e.g., Spratling, 2006, or Lücke and Sahani, 2008, for discussions). The more recently suggested approach of MCA (Lücke and Sahani, 2008) represents a fully generatively interpretable approach which achieves high reliability values. The unrestricted version of MCA_3 extracts all bars in 90% of the trials with noisy bars (with Poisson noise) and in 81% of the cases for the noiseless bars. MCA_{ET} slightly improves on these results with 92% vs. 90% for noisy bars and 82% vs. 81% in the non-noisy case (experiments with $N = 500$). If more data points are used, MCA_{ET} shows close to 100% reliability (50 of 50 trials successful, see above).

Other than the standard bars test, there has recently been an increasing interest in bars with more pronounced overlap. We therefore used a version of the bars test as suggested by Lücke (2004). For this data, bars of the same orientation can overlap (two neighboring vertical bars are not disjoint but overlap). For the test we adopted the same parameter setting for such input as used by Spratling (2006) and Lücke and Sahani (2008), that is, $N = 400$ example patterns, 16 bars, $D = 9 \times 9$, bars appear with probability $\frac{2}{16}$, and number of hidden units is $H = 32$. Bars are two pixels wide such that parallel neighboring bars have a one pixel wide region of overlap. Figure 8A shows some examples of the data points used. We applied MCA_{ET} to the data using the same parameters as for the standard bars test except of a higher initial temperature ($T = 23$) and longer cooling time (the cooling schedule in Figure 4B was stretched by a factor four to 400 iterations). In 21 of the 25 trials the system extracted all bars (see Figure 8B). In four trials 15 of the 16 bars were represented. The average number of extracted bars was thus 15.84. In all the successful trials, reconstruction of the generating parameters was high with a maximal MAE of 0.05 and a mean MAE of 0.04 (bars have

1. Note that alternatively to using $N = 10\,000$ and 200 iterations, we could, for non-noisy data, simply add Gaussian pixel noise. This would take us back to the noisy-bars test and we would obtain close to 100% reliability with $N = 1000$ data points.

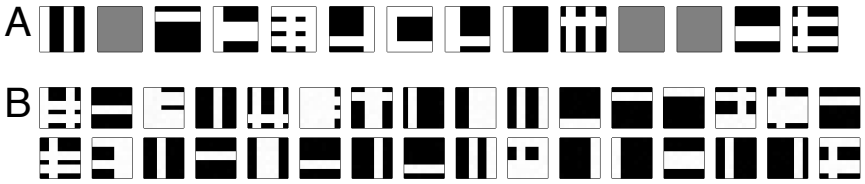


Figure 8: **A** Selection of 14 data points of a bars test with increased overlap. **B** Typical parameters W after learning if twice as many hidden units as bars are used. The supernumerous units are used to represent composite patterns.

value 10). As for the standard bars test we found that convergence to local optima was caused by effects of the finite sample size. When we repeated the experiment with the same generating and model parameters but with $N = 800$ instead of $N = 400$ data points, the algorithm extracted all bars in all of 50 trials (mean number of bars extracted equal to 16.0). In work by Spratling (2006) state-of-the-art systems were quantitatively compared using the mean number of extracted bars. From the evaluated systems only few achieved values close or equal to the optimal value of 16 for $N = 400$. From the systems with high values, many required additional constraints on the parameters (e.g., constrained forms of NMF) that had to be set by hand (see Spratling, 2006; Lücke and Sahani, 2008, for discussions).

In general, the component extraction performance of MCA_{ET} on the different bars test tasks is similar to the performance of MCA_3 suggested by Lücke and Sahani, 2008. In terms of computational cost, MCA_{ET} represents a substantial improvement, however (even compared to R-MCA_2, a constrained form of MCA; see Lücke and Sahani, 2008). This allows for applications with large H as demonstrated, for example, in the following section.

4.2.2 EXPERIMENTS - MORE REALISTIC DATA

As an example for an application to more realistic data, we applied MCA_{ET} to visual data in the form of image patches. We used the same image, Figure 9A, as in work by Lücke and Sahani (2008) to allow for a comparison. We randomly selected $N = 40\,000$ patches of 10×10 pixels as data points (see Figure 9D for ten examples). The data points were globally scaled to lie in the interval $[0, 10]$. However, just very few pixels had values close to 10 after scaling. The mean pixel value was 1.6 and thus smaller than for the bars test. We therefore used a smaller assumed Gaussian noise for the model (standard deviation $\sigma = 1.0$ instead of $\sigma = 2.0$) and started cooling at a lower temperature of $T^{\text{init}} = 4.0$. Also the small noise term on the model parameters was scaled down (0.01 instead of 0.05). During learning, we cooled for 400 iterations (cooling schedule of Figure 4B stretched by a factor four) and allowed for additional 400 iterations at T^{final} to guarantee full convergence (although changes after iteration 400 were small).

Figures 9B and 9C show the resulting parameters W after applying MCA_{ET} with $H = 50$ and $H = 100$ hidden units, respectively. For the approximations we again used $H' = 5$ and $\gamma = 3$. As can be observed, the extracted generative fields represent typical components of the training patches (compare Figure 9B-D). Data generated according to the MCA generative model using the

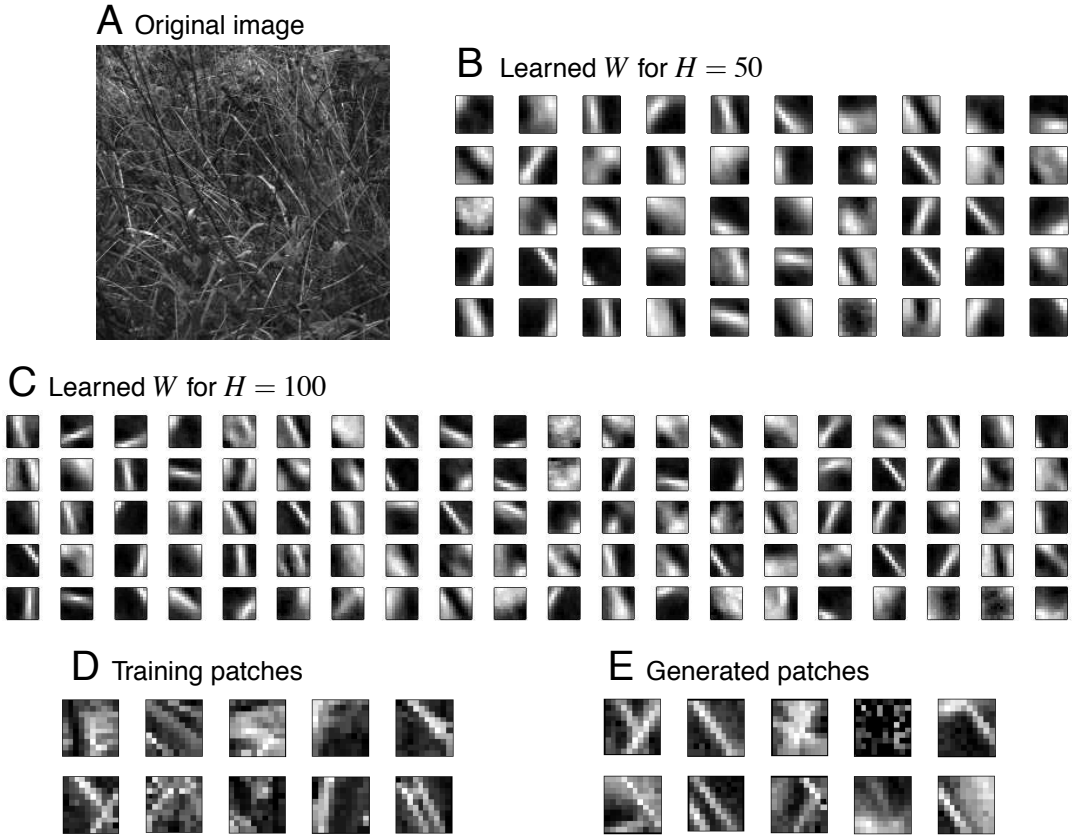


Figure 9: Application to visual data. **A** The 250-by-250 pixel image used as basis for the experiment. The image is taken from the van Hateren database of natural images (brightened for visualization). **B** Parameters $\vec{W}_h = (W_{h1}, \ldots, W_{hD})^T$ of the MCA generative model with $H = 50$ trained by ET with $H' = 5$ and $\gamma = 3$. **C** Parameters $\vec{W}_h$ of MCA with $H = 100$ ($H' = 5$ and $\gamma = 3$). **D** A selection of 10 typical training patches. **E** Ten examples of patches generated according to the MCA generative model using the generative fields in **C**. To reduce the apparent noise level, the patches were generated with a smaller σ than for training.

extracted generative fields thus resemble the structure of the training patches (Figure 9E). Note that the components of the data (e.g., images of grass blades and stems) superimpose non-linearly which motivated the application of MCA.

Due to impractically long computation times, no previous version of MCA could be applied to numbers of hidden units much larger than 50. The maximum achievable so far, was the application of the constrained MCA version (R-MCA$_2$) with $H = 50$ to $N = 5000$ patches of 10×10 pixels. As shown in the examples of Figure 9, Expectation Truncation allows for larger scale applications of unconstrained MCA with $H = 100$ and beyond. Compared to MCA$_3$ the number of states that

have to be evaluated by MCA_{ET} for $H = 100$ is reduced by more than three orders of magnitude (see Section 5.2 and Appendix C for discussions).

4.3 Linear Components Analysis (LinCA)

As third and final example let us discuss a generative model whose components can be negative as well as positive. Consider, therefore, the model given by (22), (24) and (25) without the restriction to non-negative weights. For small π in (22) such a generative model is reminiscent of sparse coding (SC; Olshausen and Field, 1996): its hidden variables are sparsely active, its basis functions are combined linearly, and its observed variables are, given the latents, independently drawn from standard Gaussian distributions. Instead of defining explicit prior and noise distributions, SC-like models are often, more informally, written in the form:

$$\vec{y} = \sum_{h=1}^{H} s_h \vec{W}_h + \sigma\vec{\eta} = W\vec{s} + \sigma\vec{\eta}, \tag{32}$$

where the entries of $\vec{\eta}$ are independently and identically drawn from a Gaussian distribution with zero mean and unit variance.

The model (32) with binary factors s_h distributed according to the Bernoulli prior (22) can be trained with Expectation Truncation. As M-step we can directly use Equation 26. However, in contrast to the previous generative models, we can not use Equation 28 as a selection function because it was derived assuming non-negative entries W_{dh}. To find appropriate selection functions, let us first consider a special case: let us assume the number of observed and hidden variables to be identical, $H = D$, and let us assume zero observation noise ($\sigma = 0$ in Equation 32; also compare Teh et al., 2003). For continuous factors s_h with non-Gaussian priors this special case represents the standard version of ICA (see, e.g., Comon, 1994; Hyvärinen et al., 2009). ICA is deterministic in the sense that for any given data point $\vec{y}^{(n)}$ the generating hidden vector $\vec{s}^{(n)}$ is known exactly. If W is invertible, the generating hidden vector is given by $\vec{s}^{(n)} = W^{-1}\vec{y}^{(n)}$ (as can directly be deduced from Equation 32 with $\sigma = 0$). ICA is most frequently applied to (PCA-)whitened data (compare, e.g., Bishop, 2006; Hyvärinen et al., 2009), in which case W is an orthogonal matrix ($W^{-1} = W^T$). For ICA on whitened data, the generating hidden units of a data point $\vec{y}^{(n)}$ are thus given by $\vec{s}^{(n)} = W^T\vec{y}^{(n)}$. In other words, the conditional distribution $p(\vec{y}\,|\,\vec{s},\Theta)$ and thus the posterior $p(\vec{s}\,|\,\vec{y}^{(n)},\Theta)$ become equal to the delta function $\delta(\vec{s} - W^T\vec{y}^{(n)})$. For $\sigma > 0$ in Equation 32, the conditional distribution $p(\vec{y}\,|\,\vec{s},\Theta)$ is, in hidden space, a Gaussian function with mean $W^T\vec{y}^{(n)}$, and the posterior is proportional to the product of this function with the prior. A multiplication with a sparse prior has the effect of moving the Gaussian function closer to those axes h with large values $\vec{W}_h^T\vec{y}^{(n)}$. The scalar products $\vec{W}_h^T\vec{y}^{(n)}$ can thus serve to select those hidden dimensions which span the space most posterior mass lies near to.

Knowing where most posterior mass is concentrated is the crucial prerequisite for finding selection functions for ET. In the binary case of model (32) with Bernoulli prior (22), selecting the hidden dimensions corresponds to selecting the hidden variables which are most likely to have non-zero entries. In analogy to the ICA case, we thus use the scalar product to define selection functions:

$$\mathcal{S}_h(\vec{y}^{(n)}) = \frac{\vec{W}_h^{\mathrm{T}}\vec{y}^{(n)}}{|\vec{W}_h|\,|\vec{y}^{(n)}|}, \quad \text{with } |\vec{v}| = \sqrt{\textstyle\sum_{i=1}^{D}(v_i)^2} \quad \text{and} \quad \vec{W}_h^T = (W_{1h},\ldots,W_{Dh}). \tag{33}$$

We use normalized scalar products because we are not constrained to an orthogonal matrix W and want to prevent the lengths of $|\vec{W}_h|$ from having a strong influence on the selections. Note, however,

that if the matrix W is orthogonal, the selection is equivalent to one based on non-normalized scalar products. The normalization by $|\vec{y}^{(n)}|$ does not affect the selection because $\vec{y}^{(n)}$ is independent of h.

Selection functions (33), M-step equation (26), and the ET approximation of the E-step represent a learning algorithm that will be referred to as Linear Components Analysis (LinCA).

4.3.1 EXPERIMENTS - ARTIFICIAL DATA

To study the LinCA learning algorithm it is first applied to artificial bars data. We use the same linear bars test setup as for the noisy bars test in Section 4.1.1 (compare Figure 3A) but invert five of the ten bars to negative values. Figure 10A shows 12 examples of the $N = 500$ data points used. The observed variables (or 'pixels') of positive bars have value 10.0, observed variables of negative bars have value -10.0, and all other observed variables have value zero. Consequently, we initialize the parameters W with positive and negative values by drawing iid from a Gaussian with zero mean and standard deviation 2.0. All other parameters (for initialization, approximation, annealing, and data generation) are chosen as in Section 4.1.1 for the NMF model. Figure 10B shows a typical time course of the parameters W for noisy data points. As can be observed, the parameters converge to the true generating causes relatively early. Figure 10C shows the likelihood values of ten trials with the same set of $N = 500$ data points. In most trials, likelihoods converged quickly to values close to the likelihood values of the generating parameters (dashed horizontal line). In some trials convergence took longer, however. Figure 10D shows the quality values $Q^{(n)}$ (Equation 21) during a typical trial. After an initially relatively low approximation quality, the quality values of the data points generated by less or equal γ causes (bright green lines) quickly increase. As previously, only these data points are finally used for learning. Data points generated by greater than γ causes (dark red lines) are finally discarded.

To measure the reliability of the system, we ran 50 trials with 50 different data sets of $N = 500$ data points. In all trials all bars were extracted. Parameter reconstruction in all the trials was high with all MAE smaller than 0.28 and a mean MAE of 0.21. For bars without noise we extract all bars in 49 trials (98%) and nine of ten bars in one trial. Due to the non-noisy data, parameter reconstruction was higher than for noisy data. The MAE of all successful trials was smaller than 0.09 and the mean MAE was 0.04. The reliability for non-noisy data increased to still higher reliability values when we cooled longer. If the cooling schedule in Figure 4B was stretched to 200 iterations, all bars were found in all of 100 trials. Alternatively, reliability increased when we increased the sample size: all bars were found in 100 of 100 trials if $N = 1000$ data points were used.

4.3.2 EXPERIMENTS - MORE REALISTIC DATA

As an example of more realistic data, we applied LinCA to sound waveforms. Sound waves have positive and negative parts and their components superimpose linearly, which is consistent with the assumptions of the LinCA model. As data we used short sound intervals obtained from recordings of ten different male voices uttering the sentence: "Don't ask me to carry an oily rag like that". Data was taken from the TIMIT database with voices sampled at 16kHz. To only learn from the spoken text, we cut off the silent initial part and the silent final part of each recorded sentence. Furthermore, we multiplied each voice recording by a different factor such that each recording filled the interval $[-5,5]$ (maximal absolute amplitude of each recording 5.0). This compensated for different sound levels of the different speakers and made the data range comparable to those of

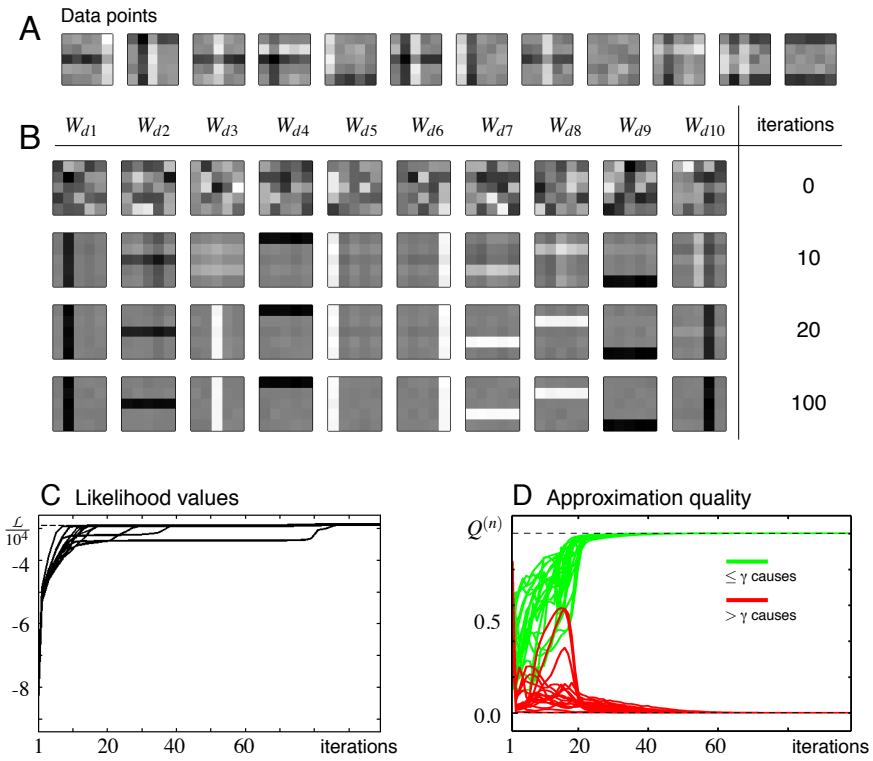


Figure 10: **A** Example data of a bars test with components of positive and negative values, $D = 5 \times 5$, and additive Gaussian noise. **B** Time course of W for the LinCA model trained with ET. **C** Likelihood values for ten trials with the same set of training patterns and different initial conditions for each trial. **D** Time course of the quality values of 40 data points during one trial. Values are plotted for twenty randomly selected data points generated by less or equal γ causes (bright green lines) and twenty randomly selected data points generated by greater than γ causes (dark red lines).

previous experiments. After the multiplication only few data values lay close to -5 or 5. As in Section 4.2.2 we therefore assumed a lower data noise than for the bars data ($\sigma = 0.5$ in this case). As data points we used all possible segments of 12.5ms length. This amounts to $N = 389\,510$ data points with $D = 200$ observed variables.

We applied LinCA with $H = 200$ hidden units to the data and used a prior parameter of $\pi = 0.01$ (on average $\pi H = 2.0$ causes per data point as in previous experiments). The approximation parameters for ET we set to $\gamma = 4$ and $H' = 10$ (as in Section 4.2, we found it beneficial to use $\overline{\mathcal{K}}_n$, which increases robustness of learning without changing the complexity; compare Appendix C). During training we annealed according to the cooling schedule in Figure 4B stretched by a factor two to 200 iterations. The initial temperature was set to $T^{\text{init}} = 5.0$ and the final temperature to $T^{\text{final}} = 1.0$. An additional 200 iterations at $T^{\text{final}} = 1$ were used to guarantee full parameter convergence (although

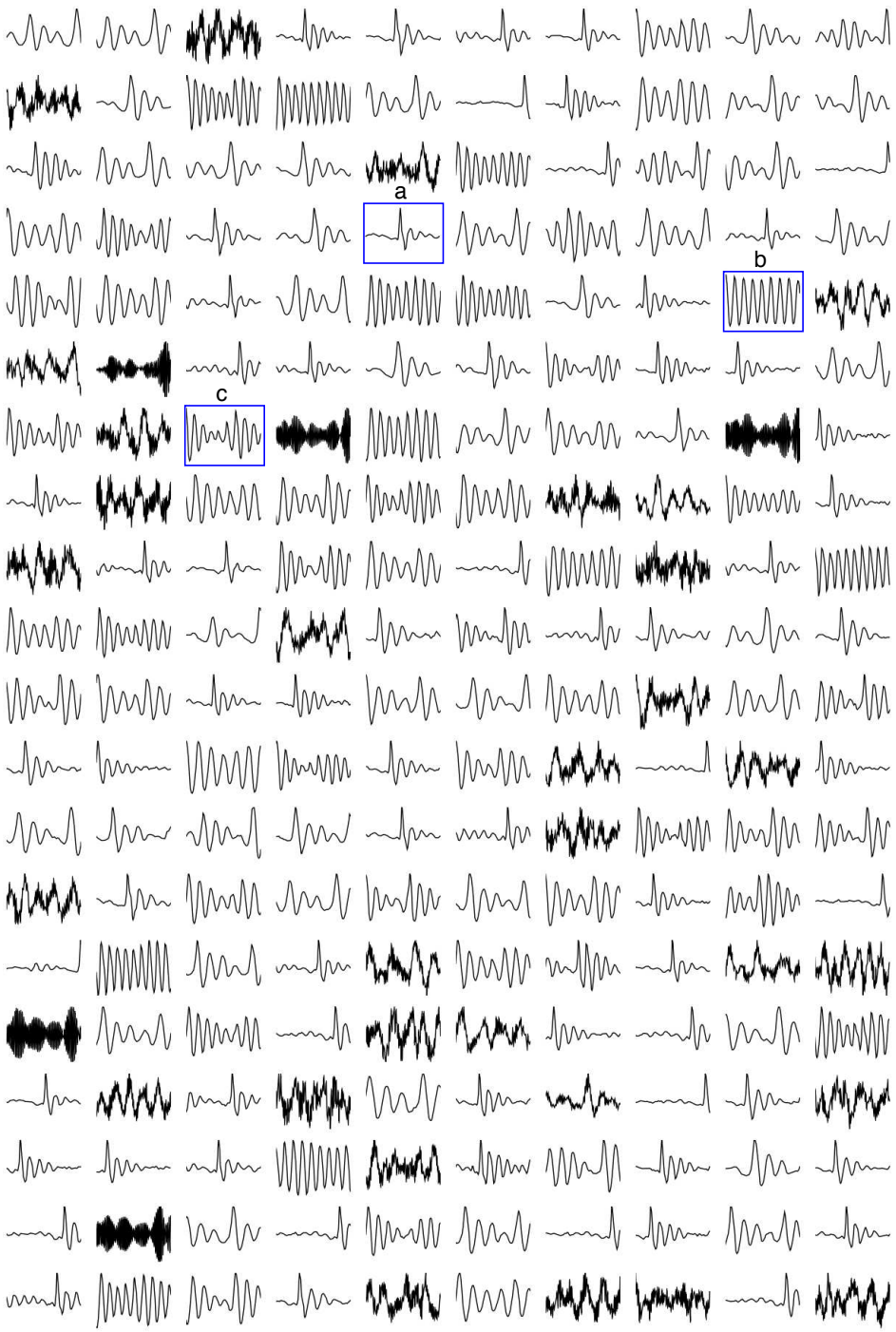


Figure 11: Parameters W after training LinCA on acoustic data. Data points were 12.5ms long intervals of voice recordings. For visualization, the $H = 200$ generative fields were individually scaled. Fields **a-c** are examples of different types of generative fields.

changes after 200 iterations were small; compare Section 4.2.2). Figure 11 shows the learnt parameters W after a typical run. Many of the generative fields (or basis functions) $\vec{W}_h$ are localized in time and frequency space (e.g., field **a**). Others are only localized in frequency space (e.g., field **b**). Still others resemble amplitude modulated waveforms (e.g., field **c**) or represent combinations of low and very high frequencies. We thus recover generative field properties similar to those obtained by other approaches (compare Teh et al., 2003, or Köster and Hyvärinen, 2007). Note, however, that we applied the algorithm to the data without preprocessing such as whitening. As for the previous experiments, no preprocessing except for data scaling was used. This is unlike SC or ICA approaches that usually require specifically preprocessed data. We also ran LinCA on a subset of the $N = 389\,510$ data points. Results with, for example, $N = 100\,000$ showed no significant differences. Similarly, a run of LinCA with $H = 400$ and $\pi H = 2.0$ showed comparable results (although we observed a tendency towards more generative fields with low absolute amplitudes in this case).

The experiments on acoustic data show that LinCA can be trained with ET also for large-scale applications. As the number of preselected candidates H' can be much smaller than the total number of hidden units, ET scales very favorably with H (see discussion in Section 5.2 and Appendix C). For the experiment of Figure 11, ET evaluates just a couple of hundreds of hidden states per data point instead of 2^{200} required for an exact E-step. Without preselection (i.e., $H' = H$) and same γ, the number of hidden states per data point is still larger than 200×10^6 (compare Appendix C), which would by far exceed currently available computational resources.

5. Discussion

We have studied an approximation to EM to train multiple-cause generative models with binary hidden variables. Training in the scheme is based on a candidate preselection to reduce the computational cost of intractable exact EM learning.

5.1 Properties of Expectation Truncation

The approximation scheme introduced in Section 2 and systematically derived in Section 3 is an example of a deterministic approximation to EM. In contrast to exact EM, there is no guarantee that the data likelihood under a given generative model is always increased or remains unchanged. In numerical experiments on a number of different generative models, we recovered, however, very close to optimal parameters for different types of data. This reflects the property of the approximation to become increasingly optimal the more it approaches the likelihood optimum. To quantify the approximation quality, we have, in different applications, monitored quality values, $Q^{(n)}$, for a selection of data points (see Section 4.1.1, Section 4.2.1, and Section 4.3.1). The quality values are themselves measures for the KL-divergence between the exact posterior (given a data point n) and its approximation (compare Section 3). The values for $Q^{(n)}$ as monitored during the experiments show that the KL-divergence for those data points, which are finally taken into account for learning, becomes virtually zero. Further away from the optimum, the approximation of ET is usually poorer than close to the optimum. However, when measured, also the initial iteration steps did increase the data likelihood in numerical experiments.

5.2 Applicability and Complexity

For the ET approximation to work, different requirements for the generative model and the data have to be fulfilled. These requirements have been made explicit in Section 3. For practical reasons, hidden variables with a small number of discrete values are the most salient requirement. In our formulation and in the experiments, we have used binary values, but ET can be applied without modifications to hidden variables with more than two discrete states. For hidden variables with continuous values the E-step can, however, not be evaluated directly (compare Equation 6 or Algorithm 2). Thus, generative models with latents of continuous values cannot be trained by the presented method. By combining ET with other approximation schemes, it could, however, be generalized to latents with continuous values. For this note that the derivation of ET in Section 3 in large parts does not rely on the assumption of discrete latents. Indeed, the derivations would still hold when all sums over $\vec{s}$ in Section 3 were replaced by integrals over the corresponding states. We are then, however, left with integrals over $\vec{s}$ (see Algorithm 2) that have to be evaluated. By applying approximation methods to these integrals over relatively small state spaces, learning algorithms for generative models with latents of continuous values could be derived.

As shown in Section 4, ET can be applied directly to data that can be generated well by combinations of binary causes. In numerical experiments we have seen examples of its application to NMF, to non-linear component superpositions (MCA), and to a form of sparse coding (LinCA). But also the applicability of ET to generative models with binary hidden variables is not without limits. From the inspection of the methods' functioning (compare Section 3) it becomes clear that it is based on the following assumption: for a sufficiently large number of data points the posterior probability mass has to be concentrated in a relatively small subset of the latents' state space. In other words, a large number of data points must be well-explicable by considering few configurations of the potential latent states. If this assumption is not fulfilled, good approximations can only be achieved by considering sums over large sets of states. We could still apply ET but the required approximation parameters would result in computational costs comparable to the ones for exact EM. Models with large numbers of hidden variables would thus not be computationally tractable in such cases. Even if the probability mass of most posteriors is finally concentrated within small regions of the state space, the applicability of ET may still be limited as it additionally requires a mechanism to locate these regions. For the generative models discussed in Section 4, the tractable selection functions (28) and (33) perform well in this respect, and successfully maximize the likelihood and recover close to optimal parameter values. However, for more complicated models the definition of tractable selection functions (or more generally of sets $\mathcal{K}_n$) may be challenging and can limit the applicability of ET. Again, the summation over more states would reduce the problem. For instance, by choosing $\mathcal{K}_n = \mathcal{K}$ (no second variational step) ET could be applied without a preselection process. However, without this crucial reduction of states, the application scale would be much more limited. In summary, applicability of ET for models with binary latents is thus not a question of principal nature but a question about the trade-off between computational cost and approximation quality.

Note that the space of models ET can be applied to is much larger than the space of models with sparsely active binary latents. ET only requires that, on average, few states can represent a data point well. These states do not have to be sparse nor do the hidden variables have to be independent. Examples are, for instance, the generative models discussed in Section 4 but with priors that fix the number of active causes (e.g., always five causes active). The requirements for ET

can still be satisfied in such situations. In preliminary numerical experiments using such non-sparse and dependent priors, ET still efficiently increased the likelihood to approximately optimal values.

In general, a reduction of the computational cost using ET is paid for by a reduction of the approximation's quality. Hereby, ET benefits from the fact that the reduction in quality is very limited compared to a vast reduction in required computations (compare Section 4). For the data considered, the computational cost can be reduced by many orders of magnitude with only minimal losses for the approximation's accuracy. For a given set of data, a good approximation setting for ET usually arises naturally. The Bernoulli prior (22), for instance, results in many data points requiring the consideration of at most πH components for good approximations. The overall computational complexity of ET thus depends on the complexity of the data and the generative model used. More specifically, it depends on how the subsets $\mathcal{K}_n$ in (6) are selected. For the generative models and data considered in this paper, a detailed discussion of ET's computational complexity can be found in Appendix C. The crucial advantage of preselection is that the computational cost is split up into essentially two parts (see Equation 38). Importantly, only the preselection part depends hereby on the total number of hidden variables H. The second and computationally more intensive part becomes independent of the total number of latents, depending only on the number of selected candidates. The preselection part is computationally much less costly: for the examples considered here, its complexity scales just linearly with H (also compare Figure 13). ET thus becomes especially efficient for large numbers of hidden variables. An example is the application of MCA_{ET} to the data of Section 4.2.2. Another example is the application of LinCA to the data of Section 4.3.2. In the latter case ET computes approximations by evaluating less than 1000 states for $H = 200$ instead of 2^{200} required for an exact E-step.

5.3 Relations to Other Approximation Schemes

A central role for optimal learning and inference in probabilistic models is played by the posterior probability $p(\vec{s}\,|\,\vec{y}^{(n)},\Theta)$. Computing expectation values w.r.t. to the posterior or computing the posterior directly is usually intractable for multiple-cause models. Approximation schemes find computationally tractable approximations to expectation values w.r.t. $p(\vec{s}\,|\,\vec{y}^{(n)},\Theta)$ or they approximate the posterior directly. Many examples of such approximations can be found in the literature, and they usually fall into two major classes: sampling methods and deterministic approaches.

Sampling methods are in a sense more general because they usually rely on fewer assumptions than deterministic approaches. Furthermore, there are no principled limits to most of them. In general, the approximations obtained recover the exact solutions in the limit of infinite computational resources. However, sampling methods can be computationally very demanding and may limit applications to relatively small scale problems. In contrast to sampling, deterministic approaches are based on analytical approximations to the posterior or its expectations computed w.r.t. it. By definition, they do in general not find the exact solutions and often rely on particular assumptions about the model they are applied to. However, if these assumptions are fulfilled, they are often computationally much less demanding. For these reasons, deterministic and sampling approaches are often regarded as complementary (see, e.g., MacKay, 2003, or Bishop, 2006, for discussions).

Major and frequently applied examples of deterministic approaches are *variational EM* approaches (e.g., Jordan et al., 1999; Jaakkola, 2000; MacKay, 2003; Bishop, 2006) and *expectation propagation* (EP) approaches (e.g., Minka, 2001; Bishop, 2006). The method of Expectation Truncation discussed in this paper is an example of a variational EM approach. Although introduced

as an approximation to expectation values (see Equation 6), we have shown in Section 3 that ET can be derived from variational approximations to exact posterior distributions. Thus, much of the typical properties of variational approaches and their differences to other approximation methods (discussed, e.g., by MacKay, 2003; Bishop, 2006) such as expectation propagation carry over to ET. A crucial difference between ET and standard variational EM is that ET does not use factorizing distributions to approximate the exact posterior. Instead ET uses two variational steps: one that takes the form of a data point selection, and a second that is based on a preselection of hidden variables (see Section 3 for more details). A consequence, for example, for the data discussed in this paper, was that we finally only learned from data points whose KL-divergence between exact and approximated posterior was virtually zero. In numerical simulations this resulted in a virtually optimal parameter recovery. At the same time, the preselection of relevant hidden variables per data point allowed a massive reduction of the computational cost.

Expectation Truncation was motivated by earlier approaches which were developed in the context of non-linear component extraction models (Lücke and Sahani, 2008; Lücke et al., 2009). For efficient learning truncated sums were used to optimize the model parameters (also see Lücke and Sahani, 2007). Optimization was partly performed under additional constraints to correct for imprecise approximations. Furthermore, the approach was developed for specific non-linear generative models and no explicit data point selection was used. As a consequence no clean relation to variational EM could be derived (but see discussion in Lücke and Sahani, 2008). Most significantly, however, no candidate preselection was used. This could result in impractically long computation times already for relatively small numbers of hidden variables. With the ET framework developed in this paper we can interpret the earlier approaches as approximations to special cases of ET. The approximations for MCA_3 (Lücke and Sahani, 2008) can thus be regarded as ET approximation without the second variational step, that is, without selecting subsets $\mathcal{K}_n$ of $\mathcal{K}$ (no preselection). Without preselection, the computational cost of learning algorithms can scale unfavorably with the number of hidden units H (cubically in the case of MCA_3; compare cases $H' = H$ in Appendix C). Data point selection in MCA_3 can be seen as implicitly accomplished by an E-step that considers more terms in the denominator than in the numerator. Likewise, the approximation for occlusive components analysis (OCA; Lücke et al., 2009) can be seen as ET without preselection (instead of implicit data point selection more terms of the truncated sums were used here). Note that instead of omitting the second variational approximation, another special case of ET is obtained if the first variational step is omitted (no data point classification). This amounts to setting $\mathcal{K}$ equal to the whole state space (the sums over $\mathcal{K}$ in Algorithm 2 equal one in this case). Efficient approximations can still be obtained if proper subsets $\mathcal{K}_n$ are selected. On different generative models preliminary experiments showed results similar to the ones reported in Section 4. Depending on the model, the approximations can be much less efficient or much less accurate, however.

An extreme case for selecting subsets $\mathcal{K}_n$ is to select them to contain just one element, $\mathcal{K}_n = \{\vec{s}^{(n)}\}$. In this case the approximated expectation value of a function g in (6) becomes equal to the functions value at $\vec{s}^{(n)}$: $\langle g(\vec{s})\rangle_{q^{(n)}} = g(\vec{s}^{(n)})$. This choice relates ET to energy models (compare, e.g., Teh et al., 2003, or Hyvärinen et al., 2009) and maximum a posteriori (MAP) approximations as, for example, used in the original sparse coding model (Olshausen and Field, 1996). Indeed, if we set $\mathcal{K}$ to be the entire state space and choose $\mathcal{K}_n = \{\vec{s}^{(n)}\}$ with $\vec{s}^{(n)}$ being the MAP estimate of the posterior, we obtain an update rule for basis functions W proportional to $\sum_n \vec{y}^{(n)} (\vec{s}^{(n)})^T$ (compare Equation 26). That is, a learning rule for W as used in sparse coding can be obtained (compare Olshausen and Field, 1996, or Olshausen, 2002). If instead of a MAP estimate the scalar

product $\vec{s}^{(n)} = \vec{W}_h^T \vec{y}^{(n)}$ is used to select $\vec{s}^{(n)}$, ET for linear generative models can be related to ICA approaches (also compare Section 4.3). Note that MAP estimate and scalar product play the role of the selection functions $\mathcal{S}_h$ in the case of SC and ICA, respectively.

More generally, MAP estimates or scalar products can be starting points to derive less basic selection functions for a generative model. Regarding a MAP estimate, it is likely that a significant amount of posterior mass is located in the vicinity of the MAP state. If the posterior is additionally known to be monomodal, $\mathcal{K}_n$ can be chosen as a region around the MAP estimate. For multi-modal posteriors, generalizations of MAP estimates could be used (see, e.g., Fromer and Globerson, 2009, and references therein). Alternatively, selection functions can be derived by considering the limit of no observation noise for a given model. For data with relatively low amounts of noise, most of the posterior mass will be located close to the estimated state for zero noise (compare the ICA case).

In numerical experiments we have compared the tractable selection functions (28) and (33) with the respective optimal selection function given by $\mathcal{S}_h^{\text{opt}}(\vec{y}^{(n)}) = p(s_h = 1 \,|\, \vec{y}^{(n)}, \Theta)$ (see Section 4.1). For low numbers of hidden variables, $p(s_h = 1 \,|\, \vec{y}^{(n)}, \Theta)$ can still be computed. Numerical experiments using the tractable selection functions, experiments using the optimal selection function, and experiments using exact EM hereby resulted in virtually identical final parameter values close to the optimum. We observed some differences in the convergence behavior. However, these differences were small compared, for example, to differences in using different annealing schemes or annealing parameters.

5.4 Results on Different Data Sets

In numerical experiments we have applied ET to three types of generative models: NMF, MCA, and LinCA. The experiments were aimed at demonstrating the method itself. However, as a by-product, we obtained some results that are closely related to recent developments in component extraction algorithms: The application of the probabilistic NMF algorithm to the MNIST data base showed that, for this data, potentially only very little is gained by considering continuous hidden units instead of binary ones. As could be observed by comparing reconstruction errors obtained for standard vs. probabilistic NMF, continuous hidden variables improved reconstructions by less than 5%. Regarding MCA, we showed that the algorithm derived with ET is competitive to the best performing systems in standard benchmarks. We applied the algorithm to the standard bars test with 10 bars and to a more recent benchmark with 16 bars and larger overlap. In both cases the algorithm extracted all causes with close to 100% reliability (50 successful trials out of 50) provided that we used sufficiently many data points. For fewer data points the algorithm was still competitive but reliability was lower. Applications of the algorithm to image patches and acoustic data showed robust applicability to large scale problems.

For all experiments in the paper we have used deterministic annealing (e.g., Ueda and Nakano, 1998; Sahani, 1999) to avoid the convergence to local optima (compare Section 4). The initial temperature for annealing can be chosen by observing that for a given model and application a critical temperature exists above which all generative fields converge to the same average field (no differentiation to different components). The initial temperature is then chosen to lie below this critical temperature. Note that for Gaussian noise, the annealing temperature changes the standard deviation σ. It is thus closely connected to the noise parameter. For many types of data, component extraction is robust to different values of σ. For instance, for the noiseless bars tests in Section 4.1,

Section 4.2, or Section 4.3, $\sigma = 2.0$ was used while the ground-truth would be $\sigma = 0.0$. For some data/model combinations the dependency on σ can be more sensitive, however.

In addition to the data and models discussed in Section 4 we also ran experiments on models with Poisson instead of Gaussian noise, used other types of data including, for example, sound spectrograms instead of sound waveforms, and used data and models in different combinations with different priors. The obtained results were all comparable to the ones reported for the other experiments: the likelihood (when computationally tractable) usually increased to close to optimal values, the generating parameters in successful trials were well recovered, and the reliability to recover the generating causes was high.

In general, the derivation of new learning algorithms based on ET is straightforward. If the data is well-explicable by combinations of binary hidden causes, a generative model should be chosen that appropriately reflects the data generation process. Once the parameter update rules for such a model are derived, the E-step can be computed with ET. This involves the definition of selection functions, the choice of an appropriate constraint for $\mathcal{K}_n$ (compare Equation 7), and the choice of approximation parameters. Depending on the data, a preselection function can take the form of an upper-bound on the joint probability as shown, for example, for NMF in Section 4.1, or it can take the form of a scalar product as for LinCA in Section 4.3. More generally, any discriminative method represents a potential choice for a selection function.

5.5 Conclusion

Motivated by earlier work that discusses the benefits of a candidate preselection (e.g., Körner et al., 1999; Lee and Mumford, 2003; Yuille and Kersten, 2006), we have defined and studied a novel approximation scheme for probabilistic generative models. This scheme is formulated as a deterministic variational EM approximation to maximize the data likelihood under a given generative model. The formulation in terms of a grounded probabilistic approach allowed us to quantify the gain in efficiency that is achievable by preselection. To study the approximation scheme empirically, it has been applied to different types of generative models with different combination rules. In standard benchmarks on artificial data we found that the derived algorithms increased the likelihood to values very close to the optimum, extracted hidden causes with high reliability, and reduced the computational cost potentially by orders of magnitude. We reported quantitative results on data with ground-truth and, where standard benchmarks were available, showed that the derived learning algorithms are competitive with the best performing systems so far. Applications to more realistic data demonstrated robustness and applicability to larger scale problems.

In conclusion, the contribution of the novel method is thus two-fold: (1) it relates the intuitive and frequently discussed benefits of preselection to the grounded framework of an EM-based approximation, and (2) it defines an approximation scheme that allows to efficiently train standard and novel types of generative models.

Acknowledgments. We would like to thank Marc Henniges for his help with the experiment of Figure 11 and Christian Keck for his help in generating animations (Online Appendix). Furthermore, we gratefully acknowledge funding by the German Research Foundation (DFG) in the project LU 1196/4-1, by the Honda Research Institute in Europe (HRI Europe), and by the German Federal Ministry of Education and Research (BMBF) in the project 01GQ0840 (BFNT Frankfurt). Finally, we would like to acknowledge support by the Frankfurt Center for Scientific Computing (CSC Frankfurt).

Appendix A. ET as Variational EM: Detailed Derivations

In Appendix A.1 we discuss the consistency of maximum likelihood parameters of original and truncated generative models. Appendix A.2 discusses details about the classification of data points.

A.1 Necessary Conditions for Global Likelihood Maxima

In Section 3.1 we have seen that $\tilde{F}(q,\Theta)$ in (15) is a lower bound of the likelihood $L(\Theta)$ in (1). If the used variational distributions $q^{(n)}(c;\Theta)$ are good approximations to the exact posteriors $p(c\,|\,\vec{y}^{\,(n)},\Theta)$ in (14), then $L(\Theta)\approx\tilde{F}(q,\Theta)$ after each E-step. Because of the variational approximation in Section 3.1 the equality $\tilde{F}(q,\Theta)=L(\Theta)$ holds if $\mathcal{M}$ is equal to $\mathcal{M}^{\text{opt}}$ (16) and if the true posterior values in (14) are equal to zero or one for all n. Although the latter condition is fulfilled only in boundary cases, we will, in this section, assume the equality to hold (while keeping in mind that it is almost always an approximation). If the equality holds, $\tilde{F}(q,\Theta)$ in (15) is in its global maximum equal to the likelihood $L(\Theta)$.

Let us, further, assume that there exist parameters Θ^* such that the original generative model reproduces the underlying distribution of the data points, $p(\vec{y})=p(\vec{y}\,|\,\Theta^*)$. From Section 3.1 we then know that the mixed model with prior (11) and (12) and $\kappa=\tilde{\kappa}$ also reproduces the original distribution for these parameters. Using the mixed model, the data points $\{\vec{y}^{\,(n)}\}_{n=1,\ldots,N}$ can thus be taken to have been generated by the truncated generative models. That is, the data set can be subdivided into the two disjoint sets $\{\vec{y}^{\,(n)}\}_{n\in\mathcal{M}^{\text{opt}}}$ and $\{\vec{y}^{\,(n)}\}_{n\notin\mathcal{M}^{\text{opt}}}$. If $p(\vec{y}\,|\,\Theta^*)$ is the underlying distribution of the whole data set, then $p(\vec{y}\,|\,c=1,\Theta^*)$ and $p(\vec{y}\,|\,c=0,\Theta^*)$ are the underlying distributions of the two disjoint parts (compare Figure 1).

We can approximately recover the distribution $p(\vec{y}\,|\,\Theta^*)$ by (globally) maximizing the data likelihood under the mixed generative model on $\{\vec{y}^{\,(n)}\}_{n=1,\ldots,N}$. Furthermore, we can recover the distributions $p(\vec{y}\,|\,c=1,\Theta^*)$ and $p(\vec{y}\,|\,c=0,\Theta^*)$ by (globally) maximizing the data likelihoods of the truncated generative models on $\{\vec{y}^{\,(n)}\}_{n\in\mathcal{M}^{\text{opt}}}$ and $\{\vec{y}^{\,(n)}\}_{n\notin\mathcal{M}^{\text{opt}}}$, respectively. Let us denote the pa-

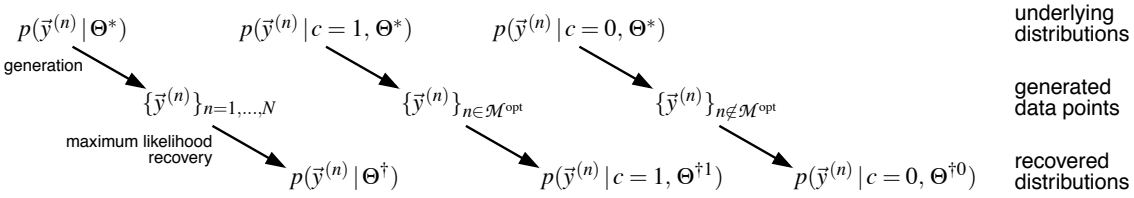


Figure 12: Recovery of the generating distributions through the original and the truncated generative models. The original distributions can be recovered from the data sets if the corresponding likelihoods are maximized.

rameters recovered by maximizing $L(\Theta)$ by $\Theta^\dagger$, and the parameters recovered by maximizing $L_1(\Theta)$ and $L_0(\Theta)$ by $\Theta^{\dagger 1}$ and $\Theta^{\dagger 0}$, respectively (compare Figure 12). In general, $\Theta^\dagger$, $\Theta^{\dagger 1}$, and $\Theta^{\dagger 0}$ are different. If the variational approximation $\mathcal{M}=\mathcal{M}^{\text{opt}}$ is exact, we know, however, that in the limit of infinitely many data points (and by still assuming $p(\vec{y})=p(\vec{y}\,|\,\Theta^*)$) applies:

$$p(\vec{y}\,|\,\Theta^*)=p(\vec{y}\,|\,\Theta^\dagger),\; p(\vec{y}\,|\,c{=}1,\Theta^*)=p(\vec{y}\,|\,c{=}1,\Theta^{\dagger 1}),\;\text{ and }\; p(\vec{y}\,|\,c{=}0,\Theta^*)=p(\vec{y}\,|\,c{=}0,\Theta^{\dagger 0}). \quad (34)$$

The equalities hold because for $N\to\infty$ and $p(\vec{y})=p(\vec{y}\,|\,\Theta^*)$ it follows from $L(\Theta^*)=L(\Theta^\dagger)$ that $D_{KL}(p(\vec{y}\,|\,\Theta^*),p(\vec{y}\,|\,\Theta^\dagger))=0$. As the Kullback-Leibler divergence between two distributions q and

p is zero if and only if the distributions are identical, (34) has to hold. Note, however, that the recovered parameters can still be different from Θ^*. For instance, if there exist transformations $\mathcal{T}$ of Θ^* that do not change the distribution, then any Θ obtained from Θ^* through such a transformation, $\Theta = \mathcal{T}(\Theta^*)$, is a likelihood maximum as well. Such multiple global maxima are the norm rather than the exception. Global maxima of models such as sparse coding (Olshausen and Field, 1996) or independent component analysis (e.g., Comon, 1994) remain global maxima under the exchange of any two basis functions or the negation of any of them.

Let all transformations $\mathcal{T}$ that map the global maxima of $L(\Theta)$ onto itself define a set that we will refer to as the *transformation set*. We say that the set of global maxima is *invariant* under the transformation set. For any two global maxima Θ^* and $\Theta^\dagger$ of $L(\Theta)$, there now exists a member $\mathcal{T}$ of the transformation set such that $\Theta^* = \mathcal{T}(\Theta^\dagger)$. Let us now demand that all global maxima of $L_1(\Theta)$ and $L_0(\Theta)$ are also invariant under the transformation set. Although this will usually be the case, for example, for the exchange of any two basis functions, it is important to state this requirement explicitly as it is not fulfilled in general. If this property is fulfilled, however, we can infer:

$$
\begin{aligned}
&\Theta^\dagger \text{ is maximum likelihood solution on } L(\Theta)\\
\Rightarrow\ &\text{There exists } \mathcal{T} \text{ such that } \Theta^\dagger = \mathcal{T}(\Theta^*) \text{ with } \Theta^* \text{ being the generating parameters.}\\
\Rightarrow\ &p(\vec{y}\,|\,c=1,\Theta^*) \text{ is the actual generating distribution of } \{\vec{y}^{(n)}\}_{n\in\mathcal{M}^{\mathrm{opt}}}\\
\Rightarrow\ &\Theta^* \text{ is maximum likelihood solution of } L_1(\Theta)\\
\Rightarrow\ &p(\vec{y}\,|\,c=1,\Theta^*) = p(\vec{y}\,|\,c=1,\mathcal{T}(\Theta^*)) = p(\vec{y}\,|\,c=1,\Theta^\dagger)\\
\Rightarrow\ &\Theta^\dagger \text{ is maximum likelihood solution of } L_1(\Theta).
\end{aligned}
$$

Analogously, $\Theta^\dagger$ is also a maximum likelihood solution of $L_0(\Theta)$ if it is a maximum likelihood solution of $L(\Theta)$. For the free-energy (15) this means that at a global maximum of $L(\Theta)$ both $L_1(\Theta)$ and $L_0(\Theta)$ also have a global maximum (under the stated assumptions). A global maximum, for example, in $L_1(\Theta)$ is thus a *necessary* condition for a global maximum in $L(\Theta)$. We have *not* shown that a maximum in $L_1(\Theta)$ is a sufficient condition for a maximum in $L(\Theta)$. Theoretically, $L_1(\Theta)$ might, for instance, not depend on all parameters, or it might have additional global maxima. Finally, note again that the necessary condition only holds under the introduced assumptions. While, for example, the assumption on invariance under transformations $\mathcal{T}$ can exactly be fulfilled (depending on the generative model), the assumptions that the true data distribution can exactly be matched or that the variational approximation in Section 3.1 is exact are in practice almost never fulfilled. The same applies for the assumption of infinitely many data points. All these assumptions can, however, be fulfilled approximately. By (globally) maximizing $L_1(\Theta)$ we can thus expect to recover parameters that maximize $L(\Theta)$ approximately.

A.2 Details of Data Classification

Starting point for choosing $\mathcal{M}$ is the set $\mathcal{M}^{\mathrm{opt}}$ in Equation 16. Setting $\mathcal{M}$=$\mathcal{M}^{\mathrm{opt}}$ would represent the best choice but without ground-truth information, $\mathcal{M}^{\mathrm{opt}}$ can not be computed exactly. We can, however, try to approximate $\mathcal{M}^{\mathrm{opt}}$. To do so, first note that we can compute an expectation value for the size of $\mathcal{M}^{\mathrm{opt}}$. It is given by $N(\mathcal{K}) = N\sum_{\vec{s}\in\mathcal{K}} p(\vec{s}\,|\,\Theta)$. We can now find an approximation to $\mathcal{M}^{\mathrm{opt}}$ by computing the values $q^{(n)}(c=1\,;\Theta^{\mathrm{old}})$ for all data points, sort them, and take the data points with the $N(\mathcal{K})$ highest values. This would represent a good approximation to $\mathcal{M}^{\mathrm{opt}}$ but it seems that we have gained very little, since we still have to compute the intractable posteriors

$q^{(n)}(c=1\,;\Theta^{\text{old}}) = p(c=1\,|\,\vec{y}^{\,(n)},\Theta^{\text{old}})$ for all n data points. Note, however, that with this procedure, the absolute values of $q^{(n)}(c=1\,;\Theta^{\text{old}})$ are not used for the approximation anymore. All that is required is a pairwise comparison of the data points based on their values $q^{(n)}(c=1\,;\Theta^{\text{old}})$.

To derive a tractable approximation of the pairwise comparison, consider two data points, $\vec{y}^{\,(n)}$ and $\vec{y}^{\,(n')}$, that are neighbors after a sorting according to $q^{(n)}(c=1\,;\Theta^{\text{old}})$. For arbitrarily many data points and for non-zero noise, the differences between the two data points become arbitrarily small. In particular, it applies for neighboring data points that the difference between the denominators of $q^{(n)}(c=1\,;\Theta^{\text{old}})$ become arbitrarily small: $\sum_{\vec{s}} p(\vec{y}^{\,(n)},\vec{s}\,|\,\Theta) \approx \sum_{\vec{s}} p(\vec{y}^{\,(n')},\vec{s}\,|\,\Theta)$. The same applies for differences between the numerators. However, as the numerators contain just small sums over $\vec{s}$, their values for neighboring data points can be expected to vary more strongly than those of the denominators. We can thus replace the comparison between $q^{(n)}(c=1\,;\Theta^{\text{old}})$ by a comparison of their numerators $\sum_{\vec{s}\in\mathcal{K}} p(\vec{y}^{\,(n)},\vec{s}\,|\,\Theta)$. This is an approximation to the pairwise comparison required for exact sorting. In the limit of infinitely many data points this procedure can be expected to result in sets $\mathcal{M}$ that represent good approximations to $\mathcal{M}^{\text{opt}}$.

If we now take preselection into account (compare Section 3.2), the comparison for sorting can be reduced further. For this note that the posterior in (14) is approximated by $q^{(n)}(c=1\,;\Theta^{\text{old}}) \approx \frac{\sum_{\vec{s}\in\mathcal{K}_n} p(\vec{y}^{\,(n)},\vec{s}\,|\,\Theta)}{\sum_{\vec{s}} p(\vec{y}^{\,(n)},\vec{s}\,|\,\Theta)}$. Following the same arguments as above, an approximation of the sorting by comparing the values $q^{(n)}(c=1\,;\Theta^{\text{old}})$ is given by sorting based on the values $\sum_{\vec{s}\in\mathcal{K}_n} p(\vec{y}^{\,(n)},\vec{s}\,|\,\Theta)$ for all n. This shows that the selection of N^{cut} data points as introduced in Section 2 (Equations 8 and 9) corresponds to defining a set $\mathcal{M}$ as an approximation to $\mathcal{M}^{\text{opt}}$.

Appendix B. Selection Function for NMF and MCA

We show that $p(s_h=1,\vec{y}^{\,(n)}\,|\,\Theta)$ is bounded by $\mathcal{S}_h(\vec{y}^{\,(n)})$ in (28) from above. This implies that $p(s_h=1\,|\,\vec{y}^{\,(n)},\Theta)$ is bounded by $\frac{\mathcal{S}_h(\vec{y}^{\,(n)})}{p(\vec{y}^{\,(n)}\,|\,\Theta)}$ from above. As $p(\vec{y}^{\,(n)}\,|\,\Theta)$ is independent of h, candidate selection based on the one bound is equivalent to selection based on the other bound. Now, consider the set $\delta_h := \{d\in\{1,\ldots,D\}\,|\,y^{(n)}_d < W_{dh}\}$ and note that if $y^{(n)}_d < W_{dh}$ and $s_h=1$ then $p(y^{(n)}_d\,|\,W_{dh},\sigma) < p(y^{(n)}_d\,|\,\overline{W}_d(\vec{s},W),\sigma)$. This is because $\overline{W}_d(\vec{s},W)$ can only be larger than W_{dh} for non-negative W and therefore the mono-modal Gaussian distribution $p(y^{(n)}_d\,|\,\overline{W}_d(\vec{s},W),\sigma)$ is further away from the maximum value. $p(y^{(n)}_d\,|\,w,\sigma)$ with $w=y^{(n)}_d$ is on the other hand larger or equal to

$p(y_d^{(n)} \,|\, w', \sigma)$ for any other value w'. It follows:

$$\begin{aligned}
p(s_h = 1, \vec{y}^{(n)} \,|\, \Theta) &= \sum_{\substack{\vec{s} \\ s_h = 1}} p(\vec{y}^{(n)} \,|\, \vec{s}, \Theta)\; p(\vec{s}\,|\,\pi) \\
&= \sum_{\substack{\vec{s} \\ s_h = 1}} \left(\prod_{d=1}^{D} p(y_d^{(n)} \,|\, \overline{W}_d(\vec{s}, W), \sigma) \right) p(\vec{s}\,|\,\pi) \\
&= \sum_{\substack{\vec{s} \\ s_h = 1}} \left(\prod_{d \in \delta_h} p(y_d^{(n)} \,|\, \overline{W}_d(\vec{s}, W), \sigma) \right) \left(\prod_{d \notin \delta_h} p(y_d^{(n)} \,|\, \overline{W}_d(\vec{s}, W), \sigma) \right) p(\vec{s}\,|\,\pi) \\
&\leq \left(\prod_{d \in \delta_h} p(y_d^{(n)} \,|\, W_{dh}, \sigma) \right) \left(\prod_{d \notin \delta_h} p(y_d^{(n)} \,|\, y_d^{(n)}, \sigma) \right) \sum_{\substack{\vec{s} \\ s_h = 1}} p(\vec{s}\,|\,\pi) \\
&= \left(\prod_{d \in \delta_h} p(y_d^{(n)} \,|\, W_{dh}^{\mathrm{ub}}, \sigma) \right) \left(\prod_{d \notin \delta_h} p(y_d^{(n)} \,|\, W_{dh}^{\mathrm{ub}}, \sigma) \right) \pi \\
&= \pi\, p(\vec{y}^{(n)} \,|\, \vec{W}_h^{\mathrm{ub}}, \sigma) \;=:\; \mathcal{S}_h(\vec{y}^{(n)}),
\end{aligned}$$

where $W_{dh}^{\mathrm{ub}} = \max\{y_d^{(n)}, W_{dh}\}$ as in Equation 28.

Appendix C. Computational Complexity

The computational cost of an E-step in the ET approximation scheme results from the computation of the selection functions $\mathcal{S}_h$ (Section 2 and Section 4) and the computation of the approximate sufficient statistics (6). For the sufficient statistics we have to compute the joint probabilities $p(\vec{s}, \vec{y}^{(n)} \,|\, \Theta)$ for different arguments. For the generative models of NMF, MCA and LinCA (Section 4.1, Section 4.2, and Section 4.3, respectively) the joint probabilities take the form:

$$p(\vec{s}, \vec{y}^{(n)} \,|\, \Theta) \;=\; \left(\prod_{d=1}^{D} p(y_d^{(n)} \,|\, \overline{W}_d(\vec{s}, W), \sigma) \right) \left(\prod_{h=1}^{H} p(s_h \,|\, \pi) \right). \tag{35}$$

For the first factor we have to compute D terms. Each term involves the computation of $\overline{W}_d(\vec{s}, W)$ which is equal to $\sum_h W_{dh} s_h$ in the cases of NMF and LinCA, and equal to $\max_h\{W_{dh} s_h\}$ in the case of MCA. In either case $\overline{W}_d(\vec{s}, W)$ can be computed with a cost proportional to the number of non-zero entries in a given vector $\vec{s}$, $\gamma' = |\vec{s}|$. The first factor in (35) is thus computable in times proportional to $D\gamma'$. The computational cost of the second factor can be neglected because just two terms (occurring γ' and $(H - \gamma')$ times) have to be computed.

The complexity of the selection functions, $\mathcal{S}_h$, used for NMF and MCA results from computing D times the conditional probabilities $p\big(y_d^{(n)} \,|\, W_{dh}^{\mathrm{ub}}, \sigma\big)$. The functions $\mathcal{S}_h$ have to be computed H times, which amounts to a computational cost proportional to DH (per data point). The selection functions used for the LinCA generative model (33) have the same computational cost. After preselection, the selection function values are used to determine the H' hidden units with the largest values, a computation that can be executed in times proportional to $\big(H + H' \log(H)\big)$ (see, e.g., Lam and Ting, 2000, for references).

In the approximate sufficient statistics (6) we sum over $|\mathcal{K}_n|$ different state vectors $\vec{s}$. If $\mathcal{K}_n$ is given as in Equation 7, it contains $\sum_{\gamma'=0}^{\gamma} \binom{H'}{\gamma'}$ different states with $\binom{H'}{\gamma'}$ counting the states with γ' non-zero entries. The denominator in (6) can therefore be computed in times proportional to $D\,\mathcal{C}(H',\gamma)$, where

$$\mathcal{C}(H',\gamma) := \sum_{\gamma'=0}^{\gamma} \binom{H'}{\gamma'} \gamma' .$$

For NMF, MCA, and LinCA the numerator can in both cases also be computed in times proportional to $D\,\mathcal{C}(H',\gamma)$ due to the special forms of the sufficient statistics. For NMF and LinCA the sufficient statistics $\langle s_i \rangle_{q^{(n)}}$ and $\langle s_i s_j \rangle_{q^{(n)}}$ do not require the computation of additional terms. For MCA the terms $\mathcal{A}^{\rho}_{di}(\vec{s},W)$ (see Equation 30) are of the form:

$$\mathcal{A}^{\rho}_{di}(\vec{s},W) = (s_i W_{di})^{\rho-1} \Big(\sum_h (s_h W_{dh})^{\rho} \Big)^{\frac{1-\rho}{\rho}} . \tag{36}$$

The last term in (36) can be computed together with $\overline{W}_d(\vec{s},W)$. Further, the computation of the D terms $p(\vec{s},\vec{y}^{(n)}\,|\,\Theta)\,\mathcal{A}_{di}(\vec{s},W)$ in the denominator is required just γ' times for a given $\vec{s}$. The numerator, like the denominator, is thus computable in times proportional to $D\gamma'$.

Selection functions and sufficient statistics have to be computed for each data point (see Algorithm 1), which results in a computational time proportional to:

$$\alpha_1 NDH \;+\; \alpha_2 N(H + H'\log(H)) \;+\; \alpha_3 ND\,\mathcal{C}(H',\gamma)\,, \tag{37}$$

where α_1, α_2, and α_3 are constants that describe potentially different weightings of the three terms. Note that if the size of $\mathcal{K}_n$ is increased by adding unit vectors, $\overline{\mathcal{K}}_n = \mathcal{K}_n \cup \{\vec{s}\,|\,\sum_i s_i = 1\}$ (as was done for MCA and LinCA) the scaling behavior (37) remains unchanged (also note that $\overline{\mathcal{K}}_n$ remains a subset of $\mathcal{K}$ as defined in Section 3.2). The usage of $\overline{\mathcal{K}}_n$ instead of $\mathcal{K}_n$ adds another $(H-H') \leq H$ terms to $\mathcal{C}(H',\gamma)$ and consequently a cost of $\alpha_4 ND(H-H') \leq \alpha_4 NDH$ to the whole expression. With a change of factor α_1, the additional cost thus gets absorbed into the first term in (37).

Considering expression (37) note that in our applications, D is always much larger than $\log(H)$. We can therefore neglect the second term such that the computational cost is approximately:[2]

$$C_{ET(H',\gamma)}(N,D,H) := \alpha_1 NDH \;+\; \alpha_3 ND\,\mathcal{C}(H',\gamma)\,. \tag{38}$$

The constants α_1 and α_3 depend on the generative model and the specific implementation of the algorithm. The principle scaling behavior remains the same, however. In Figure 13 we have plotted the scaling behavior for ET with $\alpha_1 = \alpha_3 = \frac{1}{2}$. The different graphs show the H dependence of $C_{ET(H',\gamma)}(N,D,H)$ for different values of H' and γ, $D = 100$ and $N = 1000$. Note that different choices of D and N would just globally shift the curves in the double-log plot. The steepest curve in Figure 13 is the one for ET(H,H), that is, if we choose $H' = H$ and $\gamma' = H$. In this case we drop back to the exact sufficient statistics with exponential computational cost. Note that for this

2. Note that ET also involves partial sorting of the data points once after each E-step. In a typical experiment (and in all of the experiments discussed in the paper), $\log(N)$ is much smaller than DH. The computational cost of discarding data points can therefore be neglected.

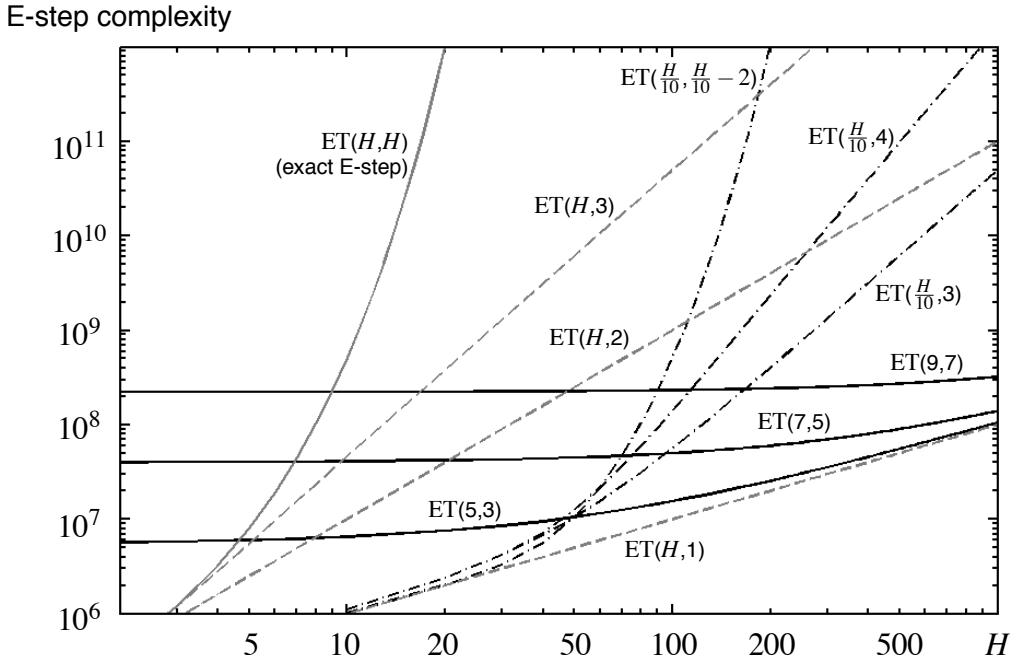


Figure 13: Scaling behavior of ET with (7) for different settings of the parameters H' and γ. The plots show the function (38) and its dependence on H.

and for all other plots with $H' = H$, we set the first term in (38) to zero because no preselection is required. The three dashed grey lines describe the computational costs of ET$(H,1)$, ET$(H,2)$, and ET$(H,3)$. They resemble straight lines because for large H their scaling is proportional to H, H^2, and H^3, respectively. Note that these three approximations are further examples for ET without preselection ($H' = H$). Their scaling behavior relative to the scaling with other parameters can therefore serve as a comparison for the computational gain of candidate preselection. The three solid black lines describe the computational time required for the parameter settings ET$(5,3)$, ET$(7,5)$, and ET$(9,7)$. These curves are flat initially because the first term in (13) (the preselection term) is small compared to the second term which is independent of H. For large H, the preselection term becomes increasingly dominant, however. The three black dashed lines describe the scaling with H for ET$(\frac{H}{10},3)$, ET$(\frac{H}{10},4)$, and ET$(\frac{H}{10},\frac{H}{10}-2)$. The computational cost of ET$(\frac{H}{10},3)$ is orders of magnitude smaller than the cost for ET$(H,3)$ but finally scales with the same slope. ET$(\frac{H}{10},4)$ is computational less expensive than ET$(H,3)$ for $H \leq 1000$ but will finally become computationally more intensive. If the parameters H' and γ' are both scaled up with H as for ET$(\frac{H}{10},\frac{H}{10}-2)$, we finally get a scaling behavior similar to that of an exact E-step. Note, however, that even in this case the computational cost remains smaller than that of ET$(H,3)$ for values of H smaller that about $H = 180$.

As discussed earlier, the computational gain achievable by ET depends on the data and the generative model used. If the data contains sufficiently many data points that are well represented by few active causes, the computational gain is potentially very substantial. For instance for the data used in this paper it was sufficient to use small values for H' and γ. Compared to systems without

preselection or an exact E-step, the gain amounts to several orders of magnitude (as, e.g., discussed for $H = 100$ in Section 4.2.2, or for $H = 200$ in Section 4.3.2).

Note that there are usually secondary effects that make learning slower if H increases. E.g., a large numbers of hidden variables usually requires large sets of data points to avoid overfitting. We also observed that longer cooling times are often beneficial for large values of D and/or H. However, ET scales essentially linearly with the number of data points, and the cooling schedule usually has to be increased merely by a factor of $2-8$. The secondary effects do, therefore, only play a role because ET has reduced the potentially exponential cost to a scaling which is close to linear in H.

Appendix D. Classical NMF vs. ET-NMF

The classical NMF belongs to the class of non-negative matrix factorization (NMF) methods, which makes use of non-negative data points, generative fields and sources. In this appendix, we first summarize the classical NMF and then derive the probabilistic version (ET-NMF) using Expectation Truncation.

D.1 Classical NMF

The standard NMF was proposed originally in Lee and Seung (1999) as a parameter-free method for the factorization of data into non-negative generative fields (or 'basis vectors') and source activities. Starting with generative fields $\vec{W}_h = (W_{1h}, \ldots, W_{Dh})^T$ for each source (or cause) s_h contained in the rows of a matrix $W \in \mathbb{R}^{D\times H}$, N data points $\vec{y}^{(n)}$ contained in the columns of a matrix $Y \in \mathbb{R}^{D\times N}$, and the corresponding source activity vectors $\vec{s}^{(n)}$ for each data point contained in the columns of a matrix $S \in \mathbb{R}^{H\times N}$, the NMF factorization tries to find an approximation

$$Y \approx WS \qquad \text{resp., in vector form} \qquad \vec{y}^{(n)} \approx W\vec{s}^{(n)} \tag{39}$$

with non-negative entries for Y, W and S, meaning that the generative fields $\vec{W}_h$ are combined linearly using the source activations $s_h^{(n)}$ to approximate the data vectors $\vec{y}^{(n)}$.

Different cost functions $E(W,S)$ have been proposed for this factorization; here we will consider an Euclidean cost function which we will compare in the next section to a Gaussian-based generative version of NMF used for Expectation Truncation (ET-NMF). To factorize (39), we therefore minimize

$$E(W,S) = ||Y - WS||^2 \tag{40}$$

with respect to W and S. Introducing

$$\langle f(\vec{y}, s_h)\rangle := \frac{1}{N}\sum_{n=1}^{N} f(\vec{y}^{(n)}, s_h^{(n)})$$

for a more compact notation and using the vectorial form of (40), the task is to minimize

$$E(W,S) = \left\langle ||\vec{y} - W\vec{s}||^2 \right\rangle , \tag{41}$$

that is, the average Euclidean reconstruction error over all data points $\vec{y}^{(n)}$.

In Lee and Seung (2001), it could be shown that the following parameter-free multiplicative update rules converge to local minima of the cost function (41),

$$s_h^{(n)} \leftarrow s_h^{(n)} \frac{(\vec{W}_h)^T \vec{y}^{(n)}}{(\vec{W}_h)^T (\sum_{h'} \vec{W}_{h'} s_{h'}^{(n)})} \quad \text{and} \quad \vec{W}_h \leftarrow \vec{W}_h \odot \frac{\langle \vec{y}\, s_h\rangle}{\sum_{h'} \vec{W}_{h'} \langle s_{h'}\, s_h\rangle}, \tag{42}$$

(the multiplication $\odot$ and division in the second equation of (42) are applied component-wise on vectors). At start, the generative fields and the source activities are initialized randomly with positive values, and then the two equations (42) are applied in alternation until convergence.

The update rules (42) conserve non-negativity and, interestingly, provide a good compromise between speed and ease of implementation. This can be seen by considering directly the gradient descent update equations derived for $\vec{W}_h$ from the minimization of Equation 41,

$$\vec{W}_h \leftarrow \vec{W}_h + \vec{\alpha} \odot \left(\langle \vec{y}\, s_h \rangle - \sum_{h'} \vec{W}_{h'} \langle s_{h'}\, s_h \rangle \right) , \tag{43}$$

with a stepsize vector $\vec{\alpha}$, and modifying an argument from Lee and Seung (2001), diagonally rescaling the variables using a stepsize

$$\vec{\alpha} = \frac{\vec{W}_h}{\sum_{h'} \vec{W}_{h'} \langle s_{h'}\, s_h \rangle} . \tag{44}$$

Inserting (44) into (43) yields the second, multiplicative update rule from (42), so that in conjunction with the convergence proof of the multiplicative NMF equations from Lee and Seung (2001) we can interprete NMF as a gradient-descent optimization of (41) with $\vec{\alpha}$ from (44) being a good choice of the gradient descent stepsize for the update of the generative fields $\vec{W}_h$.

D.2 ET-NMF

For the generative version of NMF, the choice of the source activities occurs according to the Bernoulli prior (22) from Section 4. As for the classical NMF, we combine the generative fields $\vec{W}_h$ linearly with the source activities s_h using $\overline{W}(\vec{s}, W) := W\vec{s}$, with the difference that the activities are now binary and constitute the hidden variables of the system (remark that now we write s_h instead of $s_h^{(n)}$). In addition, we use a Gaussian noise model such that the probability of a data vector $\vec{y}$ given $\vec{s}$ is defined by $p(\vec{y}\,|\,\vec{s}, \Theta)$ given by Equation 24 from Section 4.1.

At this point we apply the Expectation Maximization formalism introduced in Section 2. From the data likelihood of the generative model resp. the free energy (2) we then get for the W-relevant terms ('+...' denotes terms that do not depend on W) an expression similar to Equation 41,

$$E(W) = \left\langle ||\vec{y} - W\vec{s}||^2 \right\rangle_{\mathrm{ET}} + \ldots , \tag{45}$$

so that again the average Euclidean reconstruction error should be minimized. The difference is that now the averaging $\langle ... \rangle$ runs not only over all data points, but also over all possible source combinations $\vec{s}$ used to generate each data point. Here we wrote $\langle ... \rangle_{\mathrm{ET}}$, to express that in addition, we use the Expectation Truncation formalism for the computation of averages. That is, the averaging runs over the subset of data vectors $n \in \mathcal{M}$ and the set of source vectors $\vec{s} \in \mathcal{K}_n$ gained by the source preselection and sparseness assumptions:

$$\langle f(\vec{y}, s_h) \rangle_{\mathrm{ET}} := \sum_{n \in \mathcal{M}} \sum_{\vec{s} \in \mathcal{K}_n} p(\vec{s}|\vec{y}^{(n)}, \Theta)\, f(\vec{y}^{(n)}, s_h) = \sum_{n \in \mathcal{M}} \left\langle f(\vec{y}^{(n)}, s_h) \right\rangle_{q^{(n)}} . \tag{46}$$

From here on, all following steps can be considered in analogy to the classical NMF, by replacing $\langle ... \rangle \rightarrow \langle ... \rangle_{\mathrm{ET}}$. The task of ET-NMF is therefore to minimize (45) with respect to W, using the preselected causes and all possible source activation vectors $\vec{s}$ generated by those causes.

It can be easily seen now that the gradient-descent minimization of W and a diagonal rescaling of the stepsize according to

$$\vec{\alpha} = \frac{\vec{W}_h}{\sum_{h'} \vec{W}_{h'} \langle s_{h'} s_h \rangle_{\mathrm{ET}}}$$

can also be done in analogy to the classical NMF, leading to the multiplicative update equation for the generative fields of ET-NMF,

$$\vec{W}_h \leftarrow \vec{W}_h \odot \frac{\langle \vec{y}\, s_h \rangle_{\mathrm{ET}}}{\sum_{h'} \vec{W}_{h'} \langle s_{h'}\, s_h \rangle_{\mathrm{ET}}} \; . \tag{47}$$

The multiplicative, parameter-free equation 47 therefore leads to a near-to-optimal decrease of the ET-NMF model resp. the energy function (45) for positive data points, positive generative fields and Bernoulli-prior-distributed source activations. Since the convergence proof of the original NMF as presented by Lee and Seung (2001) considers convergence of W and S separately, it is straightforward to apply it to the ET-NMF by considering the selected source activations $\vec{s} \in \mathcal{K}_n$ in (46) as given, and applying the classical NMF to an expanded data set which incorporates the given activation vectors together with the data vectors and the probabilities as occurrence frequencies. For a fixed set of $\vec{s}$'s, (47) can then be applied until convergence.

The differences and similarities between the classical NMF and ET-NMF can then be shortly stated as follows. The classical NMF uses data points in fixed association with their corresponding continuous activation vectors and minimizes the Euclidean cost function (41) for both the source activations *and* the generative fields, the latter according to the multiplicative update rule (42). ET-NMF explores a subset of allowed binary source activations for each data point and minimizes in its M-step the Euclidean cost function (45) for the generative fields only, according to the multiplicative update rule (47). The E-step of ET-NMF is based on the truncated expectation values (6) to calculate the averaged quantities in the W update equations according to

$$\langle \vec{y}\, s_h \rangle_{\mathrm{ET}} = \sum_{n \in \mathcal{M}} \vec{y}^{\,(n)} \langle s_h \rangle_{q^{(n)}} \qquad \text{and} \qquad \langle s_{h'}\, s_h \rangle_{\mathrm{ET}} = \sum_{n \in \mathcal{M}} \langle s_{h'}\, s_h \rangle_{q^{(n)}} \tag{48}$$

so that the sufficient statistics of the ET-NMF model that have to be computed for the M-step will be given by the first and second order moments $\langle s_h \rangle_{q^{(n)}}$ and $\langle s_{h'}\, s_h \rangle_{q^{(n)}}$ of the approximate posterior. The full Expectation Truncation formalism then comprises the calculation of the expectation values $\langle s_h \rangle_{q^{(n)}}$ and $\langle s_{h'}\, s_h \rangle_{q^{(n)}}$ (needed for (48)), and afterwards the adjustment of the generative fields according to (47).

Appendix E. Details of Measurements

For the measurements in Section 4, we here give details about the procedure we used to determine if a cause is represented by the parameters of a hidden variable.

Let $\vec{y}^{\,\mathrm{cause}}_{h'}$ denote a data point showing cause h' without noise (e.g., one noiseless bar). Let further $\vec{s}_h$ denote a hidden state with just one non-zero entry at position h. To determine if cause h' is represented, we compute the approximate posterior probabilities $\tilde{p}(\vec{s}_h \,|\, \vec{y}^{\,\mathrm{cause}}_{h'}, \Theta)$ for each vector $\vec{s}_h$ using the approximation provided by ET. The hidden unit h with the highest posterior value $\tilde{p}(\vec{s}_h \,|\, \vec{y}^{\,\mathrm{cause}}_{h'}, \Theta)$ is taken as the unit representing cause h'. We only take all causes to be represented if the mapping from the causes to the representing latents is injective. Additionally, we demand

that the mean average error (MAE) between the generating cause parameters $\vec{y}_{h'}^{\,\text{cause}}$ and the model parameters $\vec{W}_h$ of the representing unit is smaller than 1.0 for each cause. For the data used in this paper, the threshold discounted any generative fields $\vec{W}_h$ with significant traces of more than one cause.

References

P. Berkes, R. E. Turner, and M. Sahani. A structured model of video reproduces primary visual cortical organisation. *PLoS Computational Biology*, 5(9):e1000495, 2009.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

N. J. Butko and J. Triesch. Learning sensory representations with intrinsic plasticity. *Neurocomputing*, 70(7-9):1130–1138, 2007.

D. Charles, C. Fyfe, D. MacDonald, and J. Koetsier. Unsupervised neural networks for the identification of minimum overcomplete basis in visual data. *Neurocomputing*, 47(1-4):119–143, 2002.

P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

P. Dayan and R. S. Zemel. Competition and multiple cause models. *Neural Computation*, 7:565 – 579, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165 – 170, 1990.

M. Fromer and A. Globerson. An LP View of the M-best MAP problem. In *Advances in Neural Information Processing Systems 22*, pages 567–575, 2009.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The 'wake-sleep' algorithm for unsupervised neural networks. *Science*, 268:1158 – 1161, 1995.

S. Hochreiter and J. Schmidhuber. Feature extraction through LOCOCODE. *Neural Computation*, 11:679 – 714, 1999.

P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing XII: Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 557 – 565. IEEE, 2002.

A. Hyvärinen, J. Hurri, and P. O. Hoyer. *Natural Image Statistics*. Springer, Heidelberg London New York, 2009.

T. Jaakkola. Tutorial on variational approximation methods. In M. Opper and D. Saad, editors, *Advanced mean field methods: theory and practice*. MIT Press, 2000.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

E. Körner, M. O. Gewaltig, U. Körner, A. Richter, and T. Rodemann. A model of computation in neocortical architecture. *Neural Networks*, 12:989 – 1005, 1999.

U. Köster and A. Hyvärinen. A two-layer ICA-like model estimated by score matching. In *Proc. International Conference on Artificial Neural Networks*, LNCS 4669, pages 798–807. Springer, 2007.

T. W. Lam and H.-F. Ting. Selecting the k largest elements with parity tests. *Discrete Appl. Math.*, 101(1-3):187–196, 2000.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91, 1999.

D. D. Lee and H. S. Seung. Algorithm for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, volume 13, 2001.

T. S. Lee and D. Mumford. Hierarchical Bayesian inference in the visual cortex. *J Opt Soc Am A Opt Image Sci Vis*, 20(7):1434–1448, 2003.

J. Lücke. Hierarchical self-organization of minicolumnar receptive fields. *Neural Networks*, 17/8–9: 1377 – 1389, 2004.

J. Lücke and M. Sahani. Generalized softmax networks for non-linear component extraction. In *Proc. International Conference on Artificial Neural Networks*, LNCS 4668, pages 657 – 667. Springer, 2007.

J. Lücke and M. Sahani. Maximal causes for non-linear component extraction. *Journal of Machine Learning Research*, 9:1227 – 1267, 2008.

J. Lücke and C. von der Malsburg. Rapid processing and unsupervised learning in a model of the cortical macrocolumn. *Neural Computation*, 16:501 – 533, 2004.

J. Lücke, R. Turner, M. Sahani, and M. Henniges. Occlusive components analysis. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1069–1077, 2009.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. Available from http://www.inference.phy.cam.ac.uk/mackay/itila/.

T. P. Minka. Expectation propagation for approximate Bayesian inference. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-800-1.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

B. A. Olshausen. *Probabilistic Models of the Brain: Perception and Neural Function*, chapter Sparse Codes and Spikes, pages 257 – 272. MIT Press, 2002.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607 – 609, 1996.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 211(11):1019 – 1025, 1999.

M. Sahani. Latent variable models for neural data analysis, 1999. PhD Thesis, Caltech.

E. Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7:51 – 71, 1995.

M. W. Spratling. Learning image components for object recognition. *Journal of Machine Learning Research*, 7:793 – 815, 2006.

Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *The Journal of Machine Learning Research*, 4(7), 2003.

N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282, 1998.

R. van Rullen and S. J. Thorpe. Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural Computation*, 13(6):1255–1283, 2001.

G. Westphal and R. P. Würtz. Combining feature- and correspondence-based methods for visual object recognition. *Neural Computation*, 21(7):1952–1989, 2009.

A. Yuille and D. Kersten. Vision as Bayesian inference: analysis by synthesis? *Trends in Cognitive Sciences*, 10(7):301–308, 2006.

Journal of Machine Learning Research 11 (2010) 2901-2934 Submitted 10/09; Revised 9/10; Published 10/10

Linear Algorithms for Online Multitask Classification

Giovanni Cavallanti CAVALLANTI@DSI.UNIMI.IT
Nicolò Cesa-Bianchi CESA-BIANCHI@DSI.UNIMI.IT
DSI, Università degli Studi di Milano
via Comelico, 39
20135 Milano, Italy

Claudio Gentile CLAUDIO.GENTILE@UNINSUBRIA.IT
DICOM, Università dell'Insubria
via Mazzini, 5
21100 Varese, Italy

Editor: Manfred Warmuth

Abstract

We introduce new Perceptron-based algorithms for the online multitask binary classification problem. Under suitable regularity conditions, our algorithms are shown to improve on their baselines by a factor proportional to the number of tasks. We achieve these improvements using various types of regularization that bias our algorithms towards specific notions of task relatedness. More specifically, similarity among tasks is either measured in terms of the geometric closeness of the task reference vectors or as a function of the dimension of their spanned subspace. In addition to adapting to the online setting a mix of known techniques, such as the multitask kernels of Evgeniou *et al.*, our analysis also introduces a matrix-based multitask extension of the p-norm Perceptron, which is used to implement spectral co-regularization. Experiments on real-world data sets complement and support our theoretical findings.

Keywords: mistake bounds, perceptron algorithm, multitask learning, spectral regularization

1. Introduction

In this work we study online supervised learning algorithms that process multiple data streams at the same time. More specifically, we consider the *multitask* classification learning problem where observed data describe different learning tasks.

Incremental multitask learning systems, which simultaneously process data from multiple streams, are widespread. For instance, in financial applications a trading platform chooses investments and allocates assets using information coming from multiple market newsfeeds. When the learning tasks are unrelated, running different instances of the same underlying algorithm, one for each task, is a sound and reasonable policy. However, in many circumstances data sources share similar traits and are therefore related in some way. Unsurprisingly, this latter situation is quite common in real-world applications. In these cases the learning algorithm should be able to capitalize on data relatedness.

In multitask classification an online linear classifier (such as the Perceptron algorithm) learns from examples associated with $K > 1$ different binary classification tasks. Our goal is to design online interacting algorithms that perform better than independent learners whenever the tasks are related. We formalize task relatedness in different ways, and derive precise formalizations of the

advantage resulting from such interaction. Our investigation considers two variants of the online multitask protocol: (1) at each time step the learner acts on a single adversarially chosen task; (2) all tasks are simultaneously processed at each time step. Each setup allows for different approaches to the multitask problem and caters for different real-world scenarios. For instance, one of the advantages of the first approach is that, in most cases, the cost of running multitask algorithms has a mild dependence on the number K of tasks. The multitask classifiers we study here manage to improve, under certain assumptions, the cumulative regret achieved by a natural baseline algorithm through the information acquired and shared across different tasks.

Our analysis builds on ideas that have been developed in the context of statistical learning where the starting point is a regularized empirical loss functional or Tikhonov functional. In that framework the objective includes a co-regularization term in the form of a squared norm in some Hilbert space of functions that favors those solutions (i.e., predictive functions for the K tasks) that lie "close" to each other. In this respect, we study two main strategies. The first approach followed here is to learn K linear functions parameterized by $u^\top = (u_1^\top, \ldots, u_K^\top) \in \mathbb{R}^{Kd}$ through the minimization of an objective functional involving the sum of a loss term plus the regularization term $u^\top A u$, where A is a positive definite matrix enforcing certain relations among tasks. Following Evgeniou et al. (2005), the K different learning problems are reduced to a single problem by choosing a suitable embedding of the input instances into a common Reproducing Kernel Hilbert Space (RKHS). This reduction allows us to solve a multitask learning problem by running any kernel-based single-task learning algorithm with a "multitask kernel" that accounts for the co-regularization term in the corresponding objective functional. We build on this reduction to analyze the performance of the Perceptron algorithm and some of its variants when run with a multitask kernel.

As described above, we also consider a different learning setup that prescribes the whole set of K learning tasks to be worked on at the same time. Once again we adopt a regularization approach, this time by adding a bias towards those solutions that lie on the same low dimensional subspace. To devise an algorithm for this model, we leverage on the well-established theory of potential-based online learners. We first define a natural extension of the p-norm Perceptron algorithm to a certain class of matrix norms, and then provide a mistake bound analysis for the multitask learning problem depending on spectral relations among different tasks. Our analysis shows a factor K improvement over the algorithm that runs K independent Perceptrons and predicts using their combined margin (see Section 1.1). The above is possible as long as the the example vectors observed at each time step are unrelated, while the sequences of multitask data are well predicted by a set of highly related linear classifiers.

1.1 Main Contributions

The contribution of this paper to the current literature is twofold. First, we provide theoretical guarantees in the form of mistake bounds for various algorithms operating within the online multitask protocol. Second, we present various experiments showing that these algorithms perform well on real problems.

Our theoretical results span across the two previously mentioned settings. In the adversarially chosen task setting, we extend the ideas introduced by Evgeniou et al. (2005) to the online learning setup, and present upper bounds which depend on task relatedness. On one hand, we show that whenever the reference vectors associated with different tasks are related, we achieve an improvement of a factor up to K over the baseline approach where K online classifiers are run in parallel and

tasks are processed in isolation. On the other hand, when tasks are unrelated our bounds become not much worse than the one achieved by separately running K classifiers. In this context, the notion of relatedness among tasks follows from the specific choice of a matrix parameter that essentially defines the so-called multitask kernel. We also provide some new insight on the role played by this kernel when used as a plug-in black-box by the Perceptron or other Perceptron-like algorithms.

In the simultaneous task setting, we introduce and analyze a matrix-based multitask extension of the p-norm Perceptron algorithm (Grove et al., 2001; Gentile, 2003) which allows us to obtain a factor K improvement in a different learning setting, where the baseline, which is still the algorithm that maintains K independent classifiers, is supposed to output K predictions per trial.

On the experimental side, we give evidence that our multitask algorithms provide a significant advantage over common baselines. In particular, we show that on a text categorization problem, where each task requires detecting the topic of a newsitem, a large multitask performance increase is attainable whenever the target topics are related. Additional experiments on a spam data set confirm the potential advantage of the p-norm Perceptron algorithm in a real-world setting.

This work is organized as follows. In Section 2 we introduce notation and formally define the *adversarially chosen task* protocol. The multitask Perceptron algorithm is presented in Section 3 where we also discuss the role of the multitask feature map and show (Section 4) that it can be used to turn online classifiers into multitask classifiers. We detail the matrix-based approach to the simultaneous multitask learning framework in Section 5. Section 6 is devoted to the theoretical analysis of a general potential-based algorithm for this setup. We conclude the paper with a number of experiments establishing the empirical effectiveness of our algorithms (Section 7).

1.2 Related Work

The problem of learning from multiple tasks has been the subject of a number of recently published papers. In Evgeniou et al. (2005) a batch multitask learning problem is defined as a regularized optimization problem and the notion of multitask kernel is introduced. More specifically, they consider a regularized functional that encodes multitask relations over tasks, thus biasing the solution of the problem towards functions that lie close to each other. Argyriou et al. (2007, 2008) build on this formalization to simultaneously learn a multitask classifier and the underlying spectral dependencies among tasks. A similar model but under cluster-based assumptions is investigated in Jacob et al. (2009). A different approach is discussed in Ando and Zhang (2005) where a structural risk minimization method is presented and multitask relations are established by enforcing predictive functions for the different tasks to belong to the same hypothesis set. Complexity results for multitask learning under statistical assumptions are also given in Maurer (2006).

In the context of online learning, multitask problems have been studied in Abernethy et al. (2007) within the learning with expert advice model. In this model the forecaster has access to a fixed set of experts and is expected to make predictions for K different tasks. Regret bounds are given under the assumption that the set of best experts for the K tasks is small, as a way to formalize task similarity. Whereas these studies consider a multitask protocol in which a single task is acted upon at each time step (what we call in this paper the adversarially chosen task protocol), the work of Lugosi et al. (2009) investigates the problem where an action for each task must be chosen at every step. The relatedness among tasks is captured by imposing restrictions on the joint action chosen at each step.

Online linear multitask algorithms for the simultaneous task setting have been studied in Dekel et al. (2007), where the separate learning tasks are collectively dealt with through a common multitask loss. Their approach, however, is fundamentally different from the one considered here. In fact, using a common loss function has more the effect of prioritizing certain tasks over the others, whereas our regularized approach hopes to benefit from the information provided by each task to speed up the learning process for the other ones. Nonetheless, it is not difficult to extend our analysis to consider a more sophisticated notion of multitask loss (see Remark 12 in Section 6.2), thus effectively obtaining a shared loss regularized multitask algorithm.

Online matrix approaches to the multitask and the related multiview learning problems were considered in various works. Matrix versions of the EG algorithm and the Winnow algorithm (related to specific instances of the quasi-additive algorithms) have been proposed and analyzed in Tsuda et al. (2005), Warmuth (2007), and Warmuth and Kuzmin (2006). When dealing with the trace norm regularizer, their algorithms could be generalized to our simultaneous multitask framework to obtain mistake bounds comparable to ours. However, unlike those papers, we do not have learning rate tuning issues and, in addition, we directly handle general nonsquare task matrices.

Finally, Agarwal et al. (2008) consider multitask problems in the restricted expert setting, where task relatedness is enforced by a group norm regularization. Their results are essentially incomparable to ours.

2. The Adversarially Chosen Task Protocol: Preliminaries

The adversarially chosen task protocol works as follows. Let K be the number of binary classification tasks indexed by $1,\dots,K$. Learning takes place in a sequential fashion: At each time step $t = 1,2,\dots$ the learner receives a task index $i_t \in \{1,\dots,K\}$ and observes an instance vector $x_t \in \mathbb{R}^d$ which plays the role of side information for the task index i_t. Based on the pair $\big(x_t, i_t\big)$ it outputs a binary prediction $\widehat{y}_t \in \{-1,1\}$ and then receives the correct label $y_t \in \{-1,1\}$ for task index i_t. So, within this scheme, the learner works at each step on a single chosen task among the K tasks and operates under the assumption that instances from different tasks are vectors of the same dimension. No assumptions are made on the mechanism generating the sequences $\big(x_1,y_1\big), \big(x_2,y_2\big), \dots$ of task examples. Moreover, similarly to Abernethy et al. (2007), the sequence of task indices $i_1, i_2, \dots$ is also generated in an adversarial manner. To simplify notation we introduce a "compound" description for the pair $\big(x_t, i_t\big)$ and denote by $\phi_t \in \mathbb{R}^{dK}$ the vector

$$\phi_t^\top \stackrel{\text{def}}{=} \big(\underbrace{0,\dots,0}_{(i_t-1)d \text{ times}} \; x_t^\top \; \underbrace{0,\dots,0}_{(K-i_t)d \text{ times}} \big) \,. \tag{1}$$

Within this protocol (studied in Sections 3 and 4) we use ϕ_t or $\big(x_t, i_t\big)$ interchangeably when referring to a multitask instance. In the following we assume instance vectors are of (Euclidean) unit norm, that is, $\|x_t\| = 1$, so that $\|\phi_t\| = 1$.

We measure the learner's performance with respect to that of a (compound) reference predictor that is allowed to use a different linear classifier, chosen in hindsight, for each one of the K tasks. To remain consistent with the notation used for multitask instances, we introduce the "compound" reference task vector $u^\top = \big(u_1^\top, \dots, u_K^\top\big)$ and define the hinge loss for the compound vector u as

$$\ell_t(u) \stackrel{\text{def}}{=} \max\big\{0, 1 - y_t\, u^\top \phi_t\big\} = \max\big\{0, 1 - y_t\, u_{i_t}^\top x_t\big\}.$$

It is understood that the compound vectors are of dimension Kd. Our goal is then to compare the learner's mistakes count to the cumulative hinge loss

$$\sum_t \ell_t(u) \tag{2}$$

suffered by the compound reference task vector u. This of course amounts to summing over time steps t the losses incurred by reference task vectors u_{i_t} with respect to instance vectors x_{i_t}.

In this respect, we aim at designing algorithms that make fewer mistakes than K independent learners when the tasks are related, and do not perform much worse than those when the tasks are completely unrelated. For instance, if we use Euclidean distance to measure task relatedness, we say that the K tasks are related if there exist reference task vectors $u_1,\ldots,u_K \in \mathbb{R}^d$ having small pairwise distances $\|u_i - u_j\|$, and achieving a small cumulative hinge loss in the sense of (2). More general notions of relatedness are investigated in the next sections.

Finally, we find it convenient at this point to introduce some matrix notation. We use I_d to refer to the $d \times d$ identity matrix but drop the subscript whenever it is clear from context. Given a matrix $M \in \mathbb{R}^{m\times n}$ we denote by $M_{i,j}$ the entry that lies in the i-th row, j-th column. Moreover, given two matrices $M \in \mathbb{R}^{m\times n}$ and $N \in \mathbb{R}^{m\times r}$ we denote by $[M,N]$ the $m \times (n+r)$ matrix obtained by the horizontal concatenation of M and N. The Kronecker or direct product between two matrices $M \in \mathbb{R}^{m\times n}$ and $N \in \mathbb{R}^{q\times r}$ is the block matrix $M \otimes N$ of dimension $mq \times nr$ whose block on row i and column j is the $q \times r$ matrix $M_{i,j}N$.

3. The Multitask Perceptron Algorithm

We first introduce a simple multitask version of the Perceptron algorithm for the protocol described in the previous section. This algorithm keeps a weight vector for each task and updates all weight vectors at each mistake using the Perceptron rule with different learning rates. More precisely, let $w_{i,t}$ be the weight vector associated with task i at time t. If we are forced (by the adversary) to predict on task i_t, and our prediction happens to be wrong, we update $w_{i_t,t-1}$ through the standard additive rule $w_{i_t,t} = w_{i_t,t-1} + \eta y_t x_t$ (where $\eta > 0$ is a constant learning rate) but, at the same time, we perform a "half-update" on the remaining $K-1$ Perceptrons, that is, we set $w_{j,t} = w_{j,t-1} + \frac{\eta}{2} y_t x_t$ for each $j \neq i_t$. This rule is based on the simple observation that, in the presence of related tasks, any update step that is good for one Perceptron should also be good for the others. Clearly, this rule keeps the weight vectors $w_{j,t}$, $j = 1,\ldots,K$, always close to each other.

The above algorithm is a special case of the *multitask Perceptron algorithm* described below. This more general algorithm updates each weight vector $w_{j,t}$ through learning rates defined by a $K \times K$ *interaction matrix* A. It is A that encodes our beliefs about the learning tasks: different choices of the interaction matrix result in different geometrical assumptions on the tasks.

The pseudocode for the multitask Perceptron algorithm using a generic interaction matrix A is given in Figure 1. At the beginning of each time step, the counter s stores the mistakes made so far plus 1. The weights of the K Perceptrons are maintained in a compound vector $w_s^\top = \left(w_{1,s}^\top,\ldots,w_{K,s}^\top\right)$, with $w_{j,s} \in \mathbb{R}^d$ for all j. The algorithm predicts y_t through the sign $\widehat{y}_t$ of the i_t-th Perceptron's margin $w_{s-1}^\top \phi_t = w_{i_t,s-1}^\top x_t$. Then, if the prediction and the true label disagree, the compound vector update rule is $w_s = w_{s-1} + \left(A \otimes I_d\right)^{-1} \phi_t$. Since $\left(A \otimes I_d\right)^{-1} = A^{-1} \otimes I_d$, the above update is equivalent to the K task updates

$$w_{j,s} = w_{j,s-1} + y_t A^{-1}_{j,i_t} x_t \qquad j = 1,\ldots,K\,.$$

Parameters: Positive definite $K \times K$ interaction matrix A.
Initialization: $w_0 = 0 \in \mathbb{R}^{Kd}$, $s = 1$.
At each time $t = 1, 2, \ldots$ do the following:

1. Observe task number $i_t \in \{1, \ldots, K\}$ and the corresponding instance vector $x_t \in \mathbb{R}^d$: $||x_t|| = 1$;
2. Build the associated multitask instance $\phi_t \in \mathbb{R}^{Kd}$;
3. Predict label $y_t \in \{-1, +1\}$ with $\widehat{y}_t = \mathrm{SGN}\big(w_{s-1}^\top \phi_t\big)$;
4. Get label $y_t \in \{-1, +1\}$;
5. If $\widehat{y}_t \neq y_t$ then update:

$$w_s = w_{s-1} + y_t \big(A \otimes I_d\big)^{-1} \phi_t, \quad s \leftarrow s + 1 \ .$$

Figure 1: The multitask Perceptron algorithm.

The algorithm is mistake-driven, hence w_{s-1} is updated (and s is increased) only when $\widehat{y}_t \neq y_t$. In the following we use $A_\otimes$ as a shorthand for $A \otimes I_d$.

We now show that the algorithm in Figure 1 has the potential to make fewer mistakes than K independent learners when the tasks are related, and does not perform much worse than that when the tasks are completely unrelated. The bound dependence on the task relatedness is encoded as a quadratic form involving the compound reference task vector u and the interaction matrix A.

We specify the online multitask problem by the sequence $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{dK} \times \{-1, 1\}$ of multitask examples.

Theorem 1 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with an interaction matrix A on any finite multitask sequence of examples $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1, 1\}$ satisfies, for all $u \in \mathbb{R}^{Kd}$,*

$$m \leq \sum_{t \in \mathcal{M}} \ell_t(u) + \max_{i=1,\ldots,K} (A^{-1})_{i,i} \big(u^\top A_\otimes u\big) + \sqrt{\max_{i=1,\ldots,K} (A^{-1})_{i,i} \big(u^\top A_\otimes u\big) \sum_{t \in \mathcal{M}} \ell_t(u)}$$

where $\mathcal{M}$ is the set of mistaken trial indices.

Theorem 1 is readily proven by using the fact that the multitask Perceptron is a specific instance of the kernel Perceptron algorithm, for example, Freund and Schapire (1999), using the so-called linear *multitask* kernel introduced in Evgeniou et al. (2005) (see also Herbster et al., 2005). This kernel is defined as follows: for any positive definite $K \times K$ interaction matrix A introduce the Kd-dimensional RKHS $\mathcal{H} = \mathbb{R}^{Kd}$ with the inner product $\langle u, v \rangle_{\mathcal{H}} = u^\top A_\otimes v$. Then define the kernel feature map $\psi : \mathbb{R}^d \times \{1, \ldots, K\} \to \mathcal{H}$ such that $\psi(x_t, i_t) = A_\otimes^{-1} \phi_t$. The kernel used by the multitask Perceptron is thus defined by

$$\mathcal{K}\big((x_s, i_s), (x_t, i_t)\big) = \big\langle \psi(x_s, i_s), \psi(x_t, i_t) \big\rangle_{\mathcal{H}} = \phi_s^\top A_\otimes^{-1} \phi_t \ . \tag{3}$$

Remark 2 *Although the multitask kernel is appealing because it makes the definition of the multitask Perceptron simple and intuitive, one easily sees that the RKHS formalism is not necessary here since the kernel is actually linear. In fact, by re-defining the feature mapping as* $\psi : \mathbb{R}^d \times \{1,\ldots,K\} \to \mathbb{R}^{Kd}$*, where* $\mathbb{R}^{Kd}$ *is now endowed with the usual Euclidean product, and by letting* $\psi(x_t,i_t) = A_\otimes^{-1/2}\phi_t$*, one gets an equivalent formulation of the multitask Perceptron based on* $A_\otimes^{-1/2}$ *rather than* $A_\otimes^{-1}$*. In the rest of the paper we occasionally adopt this alternative linear kernel formulation, in particular whenever it makes the definition of the algorithm and its analysis simpler.*

Proof [Theorem 1] We use the following version of the kernel Perceptron bound (see, e.g., Cesa-Bianchi et al., 2005),

$$m \le \sum_t \ell_t(h) + \|h\|_{\mathcal{H}}^2 \left(\max_t \|\psi(x_t,i_t)\|_{\mathcal{H}}^2\right) + \|h\|_{\mathcal{H}} \sqrt{\left(\max_t \|\psi(x_t,i_t)\|_{\mathcal{H}}^2\right) \sum_t \ell_t(h)}$$

where h is any function in the RKHS $\mathcal{H}$ induced by the kernel. The proof is readily concluded by observing that, for the kernel (3) we have

$$\|u\|_{\mathcal{H}}^2 = u^\top A_\otimes u \qquad \text{and} \qquad \|\psi(x_t,i_t)\|_{\mathcal{H}}^2 = \phi_t^\top A_\otimes^{-1}\phi_t = (A^{-1})_{i_t,i_t}$$

since ϕ_t singles out the i_t's block of matrix $A_\otimes^{-1}$. ■

In the next three subsections we investigate the role of the quadratic form $u^\top A_\otimes u$ and specialize Theorem 1 to different interaction matrices.

3.1 Pairwise Distance Interaction Matrix

The first choice of A we consider is the following simple update step (corresponding to the multitask Perceptron example we made at the beginning of this section).

$$w_{j,s} = w_{j,s-1} + \begin{cases} \frac{2}{K+1} y_t x_t & \text{if } j = i_t, \\ \frac{1}{K+1} y_t x_t & \text{otherwise.} \end{cases}$$

As it can be easily verified, this choice is given by

$$A = \begin{bmatrix} K & -1 & \ldots & -1 \\ -1 & K & \ldots & -1 \\ \ldots & \ldots & \ldots & \ldots \\ -1 & \ldots & \ldots & K \end{bmatrix} \tag{4}$$

with

$$A^{-1} = \frac{1}{K+1} \begin{bmatrix} 2 & 1 & \ldots & 1 \\ 1 & 2 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & \ldots & \ldots & 2 \end{bmatrix} .$$

We have the following result.

Corollary 3 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with the interaction matrix (4) on any finite multitask sequence of examples* $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1, 1\}$ *satisfies, for all* $u \in \mathbb{R}^{Kd}$,

$$m \leq \sum_{t \in \mathcal{M}} \ell_t(u) + \frac{2\left(u^\top A_\otimes u\right)}{K+1} + \sqrt{\frac{2\left(u^\top A_\otimes u\right)}{K+1} \sum_{t \in \mathcal{M}} \ell_t(u)}$$

where

$$u^\top A_\otimes u = \sum_{i=1}^{K} \|u_i\|^2 + \sum_{1 \leq i < j \leq K} \left\|u_i - u_j\right\|^2 .$$

In other words, running the Perceptron algorithm of Figure 1 with the interaction matrix (4) amounts to using the Euclidean distance to measure task relatedness. Alternatively, we can say that the regularization term of the regularized target functional favors task vectors $u_1, \ldots, u_K \in \mathbb{R}^d$ having small pairwise distances $\left\|u_i - u_j\right\|$.

Note that when all tasks are equal, that is when $u_1 = \cdots = u_K$, the bound of Corollary 3 becomes the standard Perceptron mistake bound (see, e.g., Cesa-Bianchi et al., 2005). In the general case of distinct u_i we have

$$\frac{2\left(u^\top A_\otimes u\right)}{K+1} = \frac{2K}{K+1} \sum_{i=1}^{K} \|u_i\|^2 - \frac{4}{K+1} \sum_{1 \leq i < j \leq K} u_i^\top u_j .$$

The sum of squares $\sum_{i=1}^{K} \|u_i\|^2$ is the mistake bound one can prove when learning K independent Perceptrons (under linear separability assumptions). On the other hand, highly correlated reference task vectors (i.e., large inner products $u_i^\top u_j$) imply a large negative second term in the right-hand side of the above expression.

3.2 A More General Interaction Matrix

In this section we slightly generalize the analysis of the previous section and consider an update rule of the form

$$w_{j,s} = w_{j,s-1} + \begin{cases} \frac{b+K}{(1+b)K}\, y_t x_t & \text{if } j = i_t, \\ \frac{b}{(1+b)K}\, y_t x_t & \text{otherwise} \end{cases}$$

where b is a nonnegative parameter. The corresponding interaction matrix is given by

$$A = \frac{1}{K} \begin{bmatrix} a & -b & \ldots & -b \\ -b & a & \ldots & -b \\ \ldots & \ldots & \ldots & \ldots \\ -b & \ldots & \ldots & a \end{bmatrix} \tag{5}$$

with $a = K + b(K-1)$. It is immediate to see that the previous case (4) is recovered by choosing $b = K$. The inverse of (5) is

$$A^{-1} = \frac{1}{(1+b)K} \begin{bmatrix} b+K & b & \ldots & b \\ b & b+K & \ldots & b \\ \ldots & \ldots & \ldots & \ldots \\ b & \ldots & \ldots & b+K \end{bmatrix} .$$

When (5) is used in the multitask Perceptron algorithm, Theorem 1 can be specialized to the following result.

Corollary 4 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with the interaction matrix (5) on any finite multitask sequence of examples $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1, 1\}$ satisfies, for all $u \in \mathbb{R}^{Kd}$,*

$$m \le \sum_{t \in \mathcal{M}} \ell_t(u) + \frac{(b+K)}{(1+b)K}\left(u^\top A_\otimes u\right) + \sqrt{\frac{(b+K)}{(1+b)K}\left(u^\top A_\otimes u\right) \sum_{t \in \mathcal{M}} \ell_t(u)}$$

where

$$u^\top A_\otimes u = \sum_{i=1}^{K} \|u_i\|^2 + bK \,\mathrm{VAR}[u]$$

being $\mathrm{VAR}[u] = \frac{1}{K}\sum_{i=1}^{K} \|u_i - \overline{u}\|^2$ *the "variance", of the task vectors, and* $\overline{u}$ *the centroid* $\frac{1}{K}\left(u_1 + \cdots + u_K\right)$.

It is interesting to investigate how the above bound depends on the trade-off parameter b. The optimal value of b (requiring prior knowledge about the distribution of $u_1, \ldots, u_K$) is

$$b = \max\left\{0, \sqrt{(K-1)\frac{\|\overline{u}\|^2}{\mathrm{VAR}[u]}} - 1\right\} .$$

Thus b grows large as the reference task vectors u_i get close to their centroid $\overline{u}$ (i.e., as all u_i get close to each other). Substituting this choice of b gives

$$\frac{(b+K)}{(1+b)K}\left(u^\top A_\otimes u\right) = \begin{cases} \|u_1\|^2 + \cdots + \|u_K\|^2 & \text{if } b = 0, \\ \left(\|\overline{u}\| + \sqrt{K-1}\sqrt{\mathrm{VAR}[u]}\right)^2 & \text{otherwise.} \end{cases}$$

When the variance $\mathrm{VAR}[u]$ is large (compared to the squared centroid norm $\|\overline{u}\|^2$), then the optimal tuning of b is zero and the interaction matrix becomes the identity matrix, which amounts to running K independent Perceptron algorithms. On the other hand, when the optimal tuning of b is nonzero we learn K reference vectors, achieving a mistake bound equal to that of learning a *single* vector whose length is $\|\overline{u}\|$ plus $\sqrt{K-1}$ times the standard deviation $\sqrt{\mathrm{VAR}[u]}$.

At the other extreme, if the variance $\mathrm{VAR}[u]$ is zero (namely, when all tasks coincide) then the optimal b grows unbounded, and the quadratic term $\frac{(b+K)}{(1+b)K}\left(u^\top A_\otimes u\right)$ tends to the average square norm $\frac{1}{K}\sum_{i=1}^{K} \|u_i\|^2$. In this case the multitask algorithm becomes essentially equivalent to an algorithm that, before learning starts, chooses one task at random and keeps referring all instance vectors x_t to that task (somehow implementing the fact that now the information conveyed by i_t can be disregarded).

3.3 Encoding Prior Knowledge

We could also pick the interaction matrix A so as to encode prior knowledge about tasks. For instance, suppose we know that only certain pairs of tasks are potentially related. We represent this knowledge in a standard way through an undirected graph $G = (V, E)$, where two vertices i and j

are connected by an edge if and only if we believe task i and task j are related. A natural choice for A is then $A = I + L$, where the $K \times K$ matrix L is the Laplacian of G, defined as

$$L_{i,j} = \begin{cases} d_i & \text{if } i = j, \\ -1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Here we denoted by d_i the degree (number of incoming edges) of node i. If we now follow the proof of Theorem 1, which holds for any positive definite matrix A, we obtain the following result.

Corollary 5 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with the interaction matrix $I+L$ on any finite multitask sequence of examples $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1,1\}$ satisfies, for all $u \in \mathbb{R}^{Kd}$,*

$$m \le \sum_{t \in \mathcal{M}} \ell_t(u) + c_G\, u^\top \big(I+L\big)_\otimes u + \sqrt{c_G\, u^\top \big(I+L\big)_\otimes u \sum_{t \in \mathcal{M}} \ell_t(u)}$$

where

$$u^\top \big(I+L\big)_\otimes u = \sum_{i=1}^{K} \|u_i\|^2 + \sum_{(i,j) \in E} \big\|u_i - u_j\big\|^2 \tag{6}$$

and $c_G = \max_{i=1,\ldots,K} \sum_{j=1}^{K} \frac{v_{j,i}^2}{1+\lambda_j}$. Here $0 = \lambda_1 < \lambda_2 \le \cdots \le \lambda_K$ are the eigenvalues of the positive semidefinite matrix L, and $v_{j,i}$ denotes the i-th component[1] of the eigenvector v_j of L associated with eigenvalue λ_j.

Proof Following the proof of Theorem 1, we just need to bound

$$\max_{i=1,\ldots,K} A^{-1}_{i,i} = \max_{i=1,\ldots,K} (I+L)^{-1}_{i,i} \ .$$

If $v_1, \ldots, v_K$ are the eigenvectors of L, then

$$(I+L)^{-1} = \sum_{j=1}^{K} \frac{v_j v_j^\top}{1+\lambda_j}$$

which concludes the proof. ■

Ideally, we would like to have $c_G = O\big(\frac{1}{K}\big)$. Clearly enough, if G is the clique on K vertices we expect to exactly recover the bound of Theorem 1. In fact, we can easily verify that the eigenvector v_1 associated with the zero eigenvalue λ_1 is $\big(K^{-1/2}, \ldots, K^{-1/2}\big)$. Moreover, it is well known that all the remaining eigenvalues are equal to K—see, for example, Hogben (2006). Therefore $c_G = \frac{1}{K} + \big(1 - \frac{1}{K}\big)\frac{1}{K+1} = \frac{2}{K+1}$. In the case of more general graphs G, we can bound c_G in terms of the smallest nonzero eigenvalue λ_2,

$$c_G \le \frac{1}{K} + \left(1 - \frac{1}{K}\right) \frac{1}{1+\lambda_2} \ .$$

The value of λ_2, known as the algebraic connectivity of G, is 0 only when the graph is disconnected. λ_2 is known for certain families of graphs. For instance, if G is a complete bipartite graph (i.e., if

1. Note that the orthonormality of the eigenvectors implies $v_{1,i}^2 + \cdots + v_{K,i}^2 = 1$ for all i.

tasks can be divided in two disjoint subsets T_1 and T_2 such that every task in T_1 is related to every task in T_2 and for both $i = 1,2$ no two tasks in T_i are related), then it is known that $\lambda_2 = \min\{|T_1|, |T_2|\}$.

The advantage of using a graph G with significantly fewer edges than the clique is that the sum of pairwise distances in (6) will contain less than $\binom{K}{2}$ terms. On the other hand, this reduction has to be contrasted to a larger coefficient c_G in front of $u^\top (I+L)_\otimes u$. This coefficient, in general, is related to the total number of edges in the graph (observe that the trace of L is exactly twice this total number). The role of prior knowledge is thus to avoid the insertion in A of edges connecting tasks that are hardly related, thus preventing the presence of large terms in the sum $u^\top (I+L)_\otimes u$.

4. Turning Perceptron-like Algorithms Into Multitask Classifiers

We now show how to obtain multitask versions of well-known classifiers by using the multitask kernel mapping detailed in Section 3.

4.1 The Multitask p-norm Perceptron Algorithm

We first consider the p-norm Perceptron algorithm of Grove et al. (2001) and Gentile (2003). As before, when the tasks are all equal we want to recover the bound of the single-task algorithm, and when the task vectors are different we want the mistake bound to increase according to a function that penalizes task diversity according to their p-norm distance.

The algorithm resembles the Perceptron algorithm and maintains its state in the compound *primal* weight vector $v_s \in \mathbb{R}^{Kd}$ where s stores the mistakes made so for (plus one). What sets the *multitask p-norm Perceptron* aside from the algorithm of Section 3 is that the prediction at time t is computed, for an arbitrary positive definite interaction matrix A, as $\text{SGN}\left(w_{s-1}^\top A_\otimes^{-1} \phi_t\right)$ where the *dual* weight vector w_{s-1} is a (one-to-one) transformation of the weight vector v_{s-1}, specifically $w_{s-1} = \nabla \frac{1}{2} \|v_{s-1}\|_p^2$, with $p \geq 2$. If a mistake occurs at time t, $v_{s-1} \in \mathbb{R}^{Kd}$ is updated using the multitask Perceptron rule, $v_s = v_{s-1} + y_t A_\otimes^{-1} \phi_t$. We are now ready to state the mistake bound for the the multitask p-norm Perceptron algorithm. In this respect we focus on a specific choices of p and A.

Theorem 6 *The number of mistakes m made by the p-norm multitask Perceptron, run with the pairwise distance matrix (4) and $p = 2\ln\max\{K,d\}$, on any finite multitask sequence of examples $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1,1\}$ satisfies, for all $u \in \mathbb{R}^{Kd}$,*

$$m \leq \sum_{t \in \mathcal{M}} \ell_t(u) + H + \sqrt{2H \sum_{t \in \mathcal{M}} \ell_t(u)}$$

where

$$H = \frac{8e^2 \ln\max\{K,d\}}{(K+1)^2} X_\infty^2 \left(\sum_{i=1}^K \left\| u_i + \sum_{j \neq i} (u_i - u_j) \right\|_1 \right)^2$$

and $X_\infty = \max_{t \in \mathcal{M}} \|x_t\|_\infty$.

Proof Let v_m be the primal weight vector after any number m of mistakes. By Taylor-expanding $\frac{1}{2}\|v_s\|_p^2$ around v_{s-1} for each $s = 1, \ldots, m$, and using the fact $y_t\, w_{s-1}^\top A_\otimes^{-1} \phi_t \leq 0$ whenever a mistake occurs at step t, we get

$$\frac{1}{2}\|v_m\|_p^2 \leq \sum_{s=1}^m D\left(v_s \| v_{s-1}\right) \tag{7}$$

where $D(v_s\|v_{s-1}) = \frac{1}{2}\left(\|v_s\|_p^2 - \|v_{s-1}\|_p^2\right) - y_t\, w_{s-1}^\top A_\otimes^{-1}\phi_t$ is the so-called Bregman divergence, that is, the error term in the first-order Taylor expansion of $\frac{1}{2}\|\cdot\|_p^2$ around vector v_{s-1}, at vector v_s.

Fix any $u \in \mathbb{R}^{Kd}$. Using the convex inequality for norms $u^\top v \le \|u\|_q \|v\|_p$ where $q = p/(p-1)$ is the dual coefficient of p (so that $\|\cdot\|_q$ is the dual norm of $\|\cdot\|_p$), and the fact that

$$u^\top A_\otimes v_s = u^\top A_\otimes v_{s-1} + y_t u^\top \phi_t \ge u^\top A_\otimes v_{s-1} + 1 - \ell_t(u),$$

one then obtains

$$\|v_m\|_p \ge \frac{u^\top A_\otimes v_m}{\|A_\otimes u\|_q} \ge \frac{m - \sum_{t\in\mathcal{M}} \ell_t(u)}{\|A_\otimes u\|_q}\ . \tag{8}$$

Combining (7) with (8) and solving for m gives

$$m \le \sum_{t\in\mathcal{M}} \ell_t(u) + \|A_\otimes u\|_q \sqrt{2\sum_{s=1}^{m} D(v_s\|v_{s-1})}\ . \tag{9}$$

Following the analysis contained in, for example, Cesa-Bianchi and Lugosi (2006), one can show that the Bregman term can be bounded as follows, for $t_s = t$,

$$D(v_s\|v_{s-1}) \le \frac{p-1}{2}\left\|A_\otimes^{-1}\phi_t\right\|_p^2 = \frac{p-1}{2}\|x_t\|_p^2 \left\|A_{\downarrow i_t}^{-1}\right\|_p^2$$

where $A_{\downarrow i_t}^{-1}$ is the i_t-th column of A^{-1}.

We now focus our analysis on the choice $p = 2\ln\max\{K,d\}$ which gives mistake bounds in the dual norms $\|u\|_1$ and $\|x_t\|_\infty$, and on the pairwise distance matrix (4). It is well known that for $p = 2\ln d$ the mistake bound of the single-task p-norm Perceptron is essentially equivalent to the one of the zero-threshold Winnow algorithm of Littlestone (1989). We now see that this property is preserved in the multitask extension. We have $\|x_t\|_p^2 \le e\,\|x_t\|_\infty^2$ and

$$\left\|A_{\downarrow i_t}^{-1}\right\|_p^2 \le e\left\|A_{\downarrow i_t}^{-1}\right\|_\infty^2 = e\left(A_{i_t,i_t}^{-1}\right)^2 = \frac{4e}{(K+1)^2}\ .$$

As for the dual norm $\|A_\otimes u\|_q$, we get

$$\|A_\otimes u\|_q^2 \le \|A_\otimes u\|_1^2 = \left(\sum_{i=1}^{K}\left\|u_i + \sum_{j\ne i}(u_i - u_j)\right\|_1\right)^2\ .$$

Substituting into (9) gives the desired result. ■

The rightmost factor in the expression for H in the statement of Theorem 6 reveals the way similarity among tasks is quantified in this case. To gain some intuition, assume the task vectors u_i are all sparse (few nonzero coefficients). Then H is small when the task vectors u_i have a common pattern of sparsity; that is, when the nonzero coordinates tend to be the same for each task vector. In the extreme case when all task vectors are equal (and not necessarily sparse), H becomes

$$\left(\frac{K}{K+1}\right)^2 \left(8e^2 \ln\max\{K,d\}\right)\left(\max_{t=1,\ldots,n}\|x_t\|_\infty\right)^2 \|u_1\|_1^2\ . \tag{10}$$

If $K \le d$ this bound is equivalent (apart from constant factors) to the mistake bound for the single-task zero-threshold Winnow algorithm of Littlestone (1989).

Parameters: Positive definite $K \times K$ interaction matrix A.
Initialization: $S_0 = \emptyset,\ v_0 = 0 \in \mathbb{R}^{Kd},\ s = 1$.
At each time $t = 1, 2, \ldots$ do the following:

1. Observe task number $i_t \in \{1, \ldots, K\}$ and the corresponding instance vector $x_t \in \mathbb{R}^d\ :\ ||x_t|| = 1$;
2. Build the associated multitask instance $\phi_t \in \mathbb{R}^{Kd}$ and compute $\widetilde{\phi}_t = \left(A \otimes I_d\right)^{-1/2} \phi_t$;
3. Predict label $y_t \in \{-1, +1\}$ with $\widehat{y}_t = \mathrm{SGN}\left(w_{s-1}^\top \widetilde{\phi}_t\right)$, where $w_{s-1} = \left(I + S_{s-1} S_{s-1}^\top + \widetilde{\phi}_t \widetilde{\phi}_t^\top\right)^{-1} v_{s-1}$;
4. Get label $y_t \in \{-1, 1\}$;
5. If $\widehat{y}_t \neq y_t$ then update:

$$v_s = v_{s-1} + y_t \widetilde{\phi}_t\ , \quad S_s = \left[S_{s-1}, \widetilde{\phi}_t\right]\ , \quad s \leftarrow s + 1\ .$$

Figure 2: The second-order multitask Perceptron algorithm.

Remark 7 *Note that for $p = 2$ our p-norm variant of the multitask Perceptron algorithm does not reduce to the multitask Perceptron of Figure 1. In order to obtain the latter as a special case of the former, we could use the fact that the multitask Perceptron algorithm is equivalent to the standard 2-norm Perceptron run on "multitask instances" $A_\otimes^{-1/2} \phi_t$—see Remark 2. One then obtains a proper p-norm generalization of the multitask Perceptron algorithm by running the standard p-norm Perceptron on such multitask instances. Unfortunately, this alternative route apparently prevents us from obtaining a bound as good as the one proven in Theorem 6. For example, when p is chosen as in Theorem 6 and all task vectors are equal, then multitask instances of the form $A_\otimes^{-1/2} \phi_t$ yield a bound K times worse than (10), which is obtained with instances of the form $A_\otimes^{-1} \phi_t$.*

Finally, we should mention that an alternative definition of the p-norm Perceptron for a related problem of predicting a labelled graph has been recently proposed in Herbster and Lever (2009).

4.2 The Multitask Second-order Perceptron Algorithm

We now turn to the second-order kernel Perceptron algorithm of Cesa-Bianchi et al. (2005). The algorithm, described in Figure 2, maintains in its internal state a matrix S (initialized to the empty matrix $\emptyset$) and a multitask Perceptron weight vector v (initialized to the zero vector). Just like in Figure 1, we use the subscript s to denote the current number of mistakes plus one. The algorithm computes a tentative (inverse) matrix

$$\left(I + S_{s-1} S_{s-1}^\top + \widetilde{\phi}_t \widetilde{\phi}_t^\top\right)^{-1}\ .$$

Such a matrix is combined with the current Perceptron vector v_{s-1} to predict the label y_t. If the prediction $\widehat{y}_t$ and the label y_t disagree both v_{s-1} and S_{s-1} get updated (no update takes place otherwise). In particular, the new matrix S_s is augmented by padding with the current vector $\widetilde{\phi}_t$. Since supports are shared, the computational cost of an update is not significantly larger than that for learning a single-task (see Subsection 4.2.1).

Theorem 8 *The number of mistakes m made by the multitask Second Order Perceptron algorithm in Figure 2, run with an interaction matrix A on any finite multitask sequence of examples $(\phi_1,y_1),(\phi_2,y_2),\ldots \in \mathbb{R}^{Kd}\times\{-1,1\}$ satisfies, for all $u\in\mathbb{R}^{Kd}$,*

$$m \le \sum_{t\in\mathcal{M}} \ell_t(u) + \sqrt{\left(u^\top A_\otimes u + \sum_{t\in\mathcal{M}} \left(u_{i_t}^\top x_t\right)^2\right)\sum_{j=1}^{m}\ln(1+\lambda_j)}$$

where $\mathcal{M}$ is the sequence of mistaken trial indices and $\lambda_1,\ldots,\lambda_m$ are the eigenvalues of the matrix whose (s,t) entry is $x_s^\top A_{i_s,i_t}^{-1} x_t$, with $s,t\in\mathcal{M}$.

Proof From the mistake bound for the kernel second-order Perceptron algorithm (Cesa-Bianchi et al., 2005) we have, for all h in $\mathcal{H}$,

$$m \le \sum_{t\in\mathcal{M}} \ell_t(h) + \sqrt{\left(\|h\|_{\mathcal{H}}^2 + \sum_{t\in\mathcal{M}} h(\phi_t)^2\right)\sum_{i=1}^{m}\ln(1+\lambda_i)}$$

where $\lambda_1,\ldots,\lambda_m$ are the eigenvalues of the kernel Gram matrix including only time steps in $\mathcal{M}$. Making the role of $A_\otimes$ explicit in the previous expression yields

$$\|u\|_{\mathcal{H}}^2 = u^\top A_\otimes u$$

and

$$\left\langle u,\psi(x_t,i_t)\right\rangle_{\mathcal{H}}^2 = \left(u^\top A_\otimes A_\otimes^{-1}\phi_t\right)^2 = \left(u_{i_t}^\top x_t\right)^2.$$

Finally, the kernel Gram matrix has elements $\mathcal{K}\big(\psi(x_s,i_s),\psi(x_t,i_t)\big) = \phi_s^\top A_\otimes^{-1}\phi_t = x_s^\top A_{i_s,i_t}^{-1} x_t$, where $s,t\in\mathcal{M}$. This concludes the proof. ■

Again, this bound should be compared to the one obtained when learning K independent tasks. As in the Perceptron algorithm, we have the complexity term $u^\top A_\otimes u$. In this case, however, the interaction matrix A also plays a role in the scale of the eigenvalues of the resulting multitask Gram matrix. Roughly speaking, when the tasks are close and A is the pairwise distance matrix, we essentially gain a factor $\sqrt{K}$ from the fact that $u^\top A_\otimes u$ is close to K times the complexity of the single task (according to the arguments in Section 3). On the other hand, the trace of the multitask Gram matrix $\left[\phi_s^\top A_\otimes^{-1}\phi_t\right]_{s,t\in\mathcal{M}} = \left[x_s^\top A_{i_s,i_t}^{-1} x_t\right]_{s,t\in\mathcal{M}}$ is about the same as the trace of the single task matrix, since the K times larger dimension of the multitask matrix is offset by the factor $1/K$ delivered by $A_\otimes^{-1}$ in $\left[\phi_s^\top A_\otimes^{-1}\phi_t\right]_{s,t\in\mathcal{M}}$ when compared to the single task Gram matrix $\left[x_s^\top x_t\right]_{s,t\in\mathcal{M}}$. So, in a sense, the spectral quantity $\sum_{j=1}^m \ln(1+\lambda_j)$ is similar to the corresponding quantity for the single task case. Putting together, unlike the first-order Perceptron, the gain factor achieved by a multitask second-order perceptron over the K independent tasks bound is about $\sqrt{K}$.

4.2.1 IMPLEMENTING THE MULTITASK SECOND-ORDER PERCEPTRON IN DUAL FORM

It is easy to see that the second-order multitask Perceptron can be run in dual form by maintaining K classifiers that share the same set of support vectors. This allows an efficient implementation that does not impose any significant overhead with respect to the corresponding single-task version.

Specifically, given some interaction matrix A the margin at time t is computed as (see Cesa-Bianchi et al., 2005, Theorem 3.3)

$$\begin{aligned} w_{s-1}^\top \widetilde{\phi}_t &= v_{s-1}^\top \left(I + S_{s-1} S_{s-1}^\top + \widetilde{\phi}_t \widetilde{\phi}_t^\top \right)^{-1} \widetilde{\phi}_t \\ &= y_s^\top \left(I + S_s^\top S_s \right)^{-1} S_s^\top \widetilde{\phi}_t \end{aligned} \tag{11}$$

where y_s is the s-dimensional vector whose first $s-1$ components are the labels y_i where the algorithm has made a mistake up to time $t-1$, and the last component is 0.

Note that replacing $I + S_s^\top S_s$ with $I + S_{s-1}^\top S_{s-1}$ in (11) does not change the sign of the prediction. The margin at time t can then be computed by calculating the scalar product between $S_s^\top \widetilde{\phi}_t$ and $y_s^\top \left(I + S_{s-1}^\top S_{s-1} \right)^{-1}$. Now, each entry of the vector $S_s^\top \widetilde{\phi}_t$ is of the form $A_{j,i_t}^{-1} x_j^\top x_t$, and thus computing $S_s^\top \widetilde{\phi}_t$ requires $O(s)$ inner products so that, overall, the prediction step requires $O(s)$ scalar multiplications and $O(s)$ inner products (independent of the number of tasks K).

On the other hand, the update step involves the computation of the vector $y_s^\top \left(I + S_s^\top S_s \right)^{-1}$. For the matrix update we can write

$$I + S_s^\top S_s = \begin{bmatrix} I + S_{s-1}^\top S_{s-1} & S_{s-1}^\top \widetilde{\phi}_t \\ \widetilde{\phi}_t^\top S_{s-1} & 1 + \widetilde{\phi}_t^\top \widetilde{\phi}_t \end{bmatrix}.$$

Using standard facts about the inverse of partitioned matrices (see, e.g., Horn and Johnson, 1985, Ch. 0), one can see that the inverse of matrix $I + S_s^\top S_s$ can be computed from the inverse of $I + S_{s-1}^\top S_{s-1}$ with $O(s)$ extra inner products (again, independent of K) and $O(s^2)$ additional scalar multiplications.

5. The Simultaneous Multitask Protocol: Preliminaries

The multitask kernel-based regularization approach adopted in the previous sections is not the only way to design algorithms for the multiple tasks scenario. As a different strategy, we now aim at measuring tasks relatedness as a function of the dimension of the space spanned by the task reference vectors. In matrix terms, this may be rephrased by saying that we hope to speed up the learning process, or reduce the number of mistakes, whenever the matrix of reference vectors is spectrally sparse. For reasons that will be clear in a moment, and in order to make the above a valid and reasonable goal for a multitask algorithm, we now investigate the problem of *simultaneously* producing multiple predictions after observing the corresponding (multiple) instance vectors. We therefore extend the traditional online classification protocol to a fully simultaneous multitask environment where at each time step t the learner observes exactly K instance vectors $x_{i,t} \in \mathbb{R}^d$, $i = 1, \ldots, K$. The learner then outputs K predictions $\widehat{y}_{i,t} \in \{-1, +1\}$ and obtains the associated labels $y_{i,t} \in \{-1, +1\}$, $i = 1, \ldots, K$. We still assume that the K example sequences are adversarially generated and that $\|x_{i,t}\| = 1$. We call this setting the *simultaneous multitask setting*.

Once again the underlying rationale here is that one should be able to improve the performance over the baseline by leveraging the additional information conveyed through multiple instance vectors made available all at once, provided that the tasks to learn share common characteristics. Theoretically, this amounts to postulating the existence of K vectors $u_1,\ldots,u_K$ such that *each* u_i is a good linear classifier for the corresponding sequence $(x_{i,1},y_{i,1}),(x_{i,2},y_{i,2}),\ldots$ of examples. As before, the natural baseline is the algorithm that simultaneously run K independent Perceptron algorithms, each one observing its own sequence of examples and acting in a way that is oblivious to the instances given as input to and the labels observed by its peers. Of course, we now assume that this baseline outputs K predictions per trial. The expected performance of this algorithm is simply K times the one of a single Perceptron algorithm. An additional difference that sets the protocol and the algorithms discussed here apart from the ones considered in the previous sections is that the cumulative count of mistakes is not only over time but also over tasks; that is, at each time step more the one mistake might occur since $K > 1$ predictions are output.

In the next section we show that simultaneous multitask learning algorithms can be designed in such a way that the cumulative number of mistakes is, in certain relevant cases, provably better than the K independent Perceptron algorithm. The cases where our algorithms outperform the latter are exactly those when the overall information provided by the different example sequences are related, that is, when the reference vectors associated with different tasks are "similar", while the instance vectors received during each time step are unrelated (and thus overall more informative). These notions of similarity and unrelatedness among reference and instance vectors will be formally defined later on.

5.1 Notation and Definitions

We denote by $\langle M,N\rangle = \mathrm{TR}\big(M^\top N\big)$, for $M,N \in \mathbb{R}^{d\times K}$ the Frobenious matrix inner product. Let $r = \min\{d,K\}$ and define the function $\sigma : \mathbb{R}^{d\times K} \to \mathbb{R}^r$ such that $\sigma(M) = \big(\sigma_1(M),\ldots,\sigma_r(M)\big)$, where $\sigma_1(M) \geq \cdots \geq \sigma_r(M) \geq 0$ are the singular values of a matrix $M \in \mathbb{R}^{d\times K}$. In the following, we simply write σ_i instead of $\sigma_i(M)$ whenever the matrix argument is clear from the context.

Following Horn and Johnson (1991) we say that a function $f : \mathbb{R}^r \to \mathbb{R}$ is a *symmetric gauge function* if it is an absolute norm on $\mathbb{R}^r$ and is invariant under permutation of the components of its argument. We consider matrix norms of the form $\|\cdot\| : \mathbb{R}^{d\times K} \to \mathbb{R}$ such that $\|\cdot\| = f\circ\sigma$ where f is symmetric gauge function. A matrix norm is said unitarily (orthogonally, indeed, since we only consider matrices with real entries) invariant if $\|UAV\| = \|A\|$ for any matrix A and for any unitary (orthogonal) matrices U and V for which UAV is defined. It is well known that a matrix norm is unitarily invariant if and only if it is a symmetric gauge function of the singular values of its argument.

One important class of unitarily invariant norms is given by the Schatten p-norms, $\|U\|_{s_p} \stackrel{\text{def}}{=} \|\sigma(U)\|_p$, where the right-hand expression involves a vector norm. Note that the Schatten 2-norm is the Frobenius norm, while for $p = 1$ the Schatten p-norm becomes the trace norm $\|U\|_{s_1} = \|\sigma(U)\|_1$, which is a good proxy for the rank of U, $\|\sigma(U)\|_0$.

Let M be a matrix of size $d\times K$. We denote by $\mathrm{VEC}(M)$ the vector of size Kd obtained by stacking the columns of M one underneath the other. Important relationships can be established among the Kronecker product, the VEC operator and the trace operator. In particular, we have

$$\mathrm{VEC}(MNO) \quad = \quad (O^\top \otimes M)\mathrm{VEC}(N) \tag{12}$$

for any M,N,O for which MNO is defined, and

$$\textsc{vec}(M)^\top \textsc{vec}(N) \quad = \quad \textsc{tr}(M^\top N) \tag{13}$$

for any M,N of the same order. We denote by T_{K^2} the $K^2 \times K^2$ commutation matrix such that $T_{K^2}\textsc{vec}(M) = \textsc{vec}(M^\top)$. We recall that T_{K^2} also satisfies $T_{K^2}(M \otimes N) = (M \otimes N)T_{K^2}$ for any $M,N \in \mathbb{R}^{d\times K}$.

We rely on the notation introduced by Magnus and Neudecker (1999) to derive calculus rules for functions defined over matrices. Given a differentiable function $F : \mathbb{R}^{m\times p} \to \mathbb{R}^{n\times q}$, we define the Jacobian of F at M as the matrix $\nabla F(M) \in \mathbb{R}^{nq\times mp}$

$$\nabla F(X) = \frac{\partial \textsc{vec}\big(F(M)\big)}{\partial \textsc{vec}(M)^\top} \ . \tag{14}$$

It is easy to see that (14) generalizes the well-known definition of Jacobian for vector valued functions of vector variables. The following rules, which hold for any matrix $M \in R^{K\times K}$, can be seen as extensions of standard vector derivation formulas

$$\begin{aligned} \nabla \textsc{tr}(M^p) \quad &= \quad p\,\textsc{vec}(M^{p-1})^\top \qquad\qquad p = 1,2,\ldots & (15)\\ \nabla M^\top M \quad &= \quad (I_{K^2} + T_{K^2})(I_K \otimes M^\top) \ . & (16) \end{aligned}$$

6. The Potential-based Simultaneous Multitask Classifier

As discussed in Section 5, a reasonable way to quantify the similarity among reference vectors, as well as the unrelatedness among example vectors, is to arrange such vectors into matrices, and then deal with special properties of these matrices. In order to focus on this concept, we lay out vectors as columns of $d \times K$ matrices and extend the dual norm analysis of Subsection 4.1 to matrices. The idea is to design a classifier which is able to perform much better than the K independent Perceptron baseline discussed in Section 5 whenever the set of reference vectors $u_i \in \mathbb{R}^d$ (arranged into a $d \times K$ reference matrix U), have some matrix-specific, for example, *spectral*, properties.

Our potential-based matrix algorithm for classification shown in Figure 3 generalizes the classical potential-based algorithms operating on vectors to simultaneous multitask problems with matrix examples. This family of potential-based algorithms has been introduced in the learning literature by Kivinen and Warmuth (2001) and Grove et al. (2001), and by Nemirovski and Yudin (1978) and Beck and Teboulle (2003) in the context of nonsmooth optimization. The algorithm maintains a $d \times K$ matrix W. Initially, W_0 is the zero matrix. If $s-1$ updates have been made in the first $t-1$ time steps, then the K predictions at time t are $\textsc{sgn}\big(w_{i,s-1}^\top x_{i,t}\big)$, $i = 1,\ldots,K$, where the vector $w_{i,s-1} \in \mathbb{R}^d$ is the i-th column of the the $d \times K$ matrix W_s and $x_{i,t} \in \mathbb{R}^d$ is the instance vector associated with the i-th task at time t. An update is performed if at least one mistake occurs. When the s-th update occurs at time t then W_s is computed as

$$W_s = \nabla \frac{1}{2}\,\|V_s\|^2$$

where, in turn, the columns of the $d \times K$ matrix V_s are updated using the Perceptron rule,[2] $v_{i,s} = v_{i,s-1} + y_{i,t}\,x_{i,t}\,\mathbb{1}_{\{\widehat{y}_{i,t}\neq y_{i,t}\}}$ which, as in the basic Perceptron algorithm, is mistake driven. In other

2. Here and throughout this section, $\mathbb{1}_{\{\widehat{y}_{i,t}\neq y_{i,t}\}}$ denotes the indicator function which is 1 if the label associated with the i-th task is wrongly predicted at time t, and 0 otherwise.

Parameters: Unitarily invariant norm $\|\cdot\|$.
Initialization: $V_0 = [v_{0,0}, \ldots, v_{K,0}] = 0$, $W_0 = [w_{0,0}, \ldots, w_{K,0}] = \nabla \frac{1}{2} \|V_0\|^2$, $s = 1$.
At each time $t = 1, 2, \ldots$ do the following:

1. Get multitask instance vectors $x_{1,t}, \ldots, x_{K,t} \in \mathbb{R}^d$;
2. Predict labels $y_{i,t} \in \{-1, +1\}$ with $\widehat{y}_{i,t} = \text{SGN}(w_{i,s-1}^\top x_{i,t}), \qquad i = 1, \ldots, K$;
3. Get labels $y_{i,t} \in \{-1, 1\}, \qquad i = 1, \ldots, K$;
4. If $\widehat{y}_{i,t} \neq y_{i,t}$ for some i then update:

$$
\begin{aligned}
v_{i,s} &= v_{i,s-1} + y_{i,t} x_{i,t} \mathbb{1}_{\{\widehat{y}_{i,t} \neq y_{i,t}\}} \qquad i = 1, \ldots, K \\
W_s &= \nabla \frac{1}{2} \|V_s\|^2 \\
s &\leftarrow s + 1 \, .
\end{aligned}
$$

Figure 3: The potential-based matrix algorithm for the simultaneous multitask setting.

words, the i-th column in V_{s-1} is updated if and only if the label associated with the i-th task was wrongly predicted. We say that W_s is the dual matrix weight associated with the primal matrix weight V_s. So far we left the mapping from V_s to W_s partially unspecified since we did not say anything other than it is the gradient of some unitarily invariant (squared) norm.

6.1 Analysis of Potential-based Matrix Classifiers

We now develop a general analysis of potential-based matrix algorithms for (simultaneous) multitask classification. Then, in Section 6.2 we specialize it to Schatten p-norms. The analysis proceeds along the lines of the standard proof for potential-based algorithms. Before turning to the details, we introduce a few shorthands. Let $\mathbb{1}_{i,t} = \mathbb{1}_{\{\widehat{y}_{i,t} \neq y_{i,t}\}}$, and $\mathbb{1}_t$ be the K-dimensional vector whose i-th component is $\mathbb{1}_{i,t}$. Also, e_i denotes the i-th vector of the standard basis for $\mathbb{R}^K$. Finally, we define the matrix $M_t = \sum_{i=1}^K \mathbb{1}_{i,t} y_{i,t} x_{i,t} e_i^\top$ whose i-th column is the example vector $y_{i,t} x_{i,t}$ if the label $y_{i,t}$ was wrongly predicted at time t, or the null vector otherwise. It is easy to see that, by using this notation, the update of the primal weight matrix V can be written as $V_s = V_{s-1} + M_t$.

Let $\mathcal{M}$ be the set of trial indices where at least one mistake occurred over the K tasks, and set $m = |\mathcal{M}|$. We start by Taylor-expanding $\frac{1}{2} \|V_s\|^2$ around V_{s-1} for each $s = 1, \ldots, m$ and obtain

$$
\frac{1}{2} \|V_m\|^2 \leq \sum_{s=1}^m D(V_s \| V_{s-1}) \tag{17}
$$

where $D(V_s\|V_{s-1}) = \frac{1}{2}\big(\|V_s\|^2 - \|V_{s-1}\|^2\big) - \langle W_{s-1}, M_t\rangle$ is the matrix Bregman divergence associated with $\nabla\frac{1}{2}\|\cdot\|^2$. The upper bound in (17) follows from

$$\langle W_{s-1}, M_t\rangle = \mathrm{TR}(W_{s-1}^\top M_t) = \sum_{i=1}^{K} \mathbb{1}_{i,t}\, y_{i,t}\, w_{s-1}^\top x_{i,t} \le 0$$

the last inequality holding because $\mathbb{1}_{i,t}$ is 1 if only if a mistake in the prediction for the i-th task occurs at time t.

Fix any $d \times K$ comparison matrix U and denote by $\|\cdot\|_*$ the matrix dual norm. By the convex inequality for matrix norms we have $\|V_m\|\,\|U\|_* \ge \langle V_m, U\rangle$, where $\|U\|_* = f^*\big(\sigma(U)\big)$ and f^* is the Legendre dual of function f—see Lewis (1995, Theorem 2.4). From $\langle U, V_s\rangle = \langle U, V_{s-1}\rangle + \langle U, M_t\rangle$. we obtain

$$\|V_m\| \ge \frac{\langle U, V_m\rangle}{\|U\|_*} \ge \frac{\sum_{t\in\mathcal{M}} \|\mathbb{1}_t\|_1 - \sum_{t\in\mathcal{M}} \ell_t^{\mathbb{1}}(U)}{\|U\|_*}$$

where

$$\ell_t^{\mathbb{1}}(U) \stackrel{\text{def}}{=} \sum_{i=1}^{K} \mathbb{1}_{i,t}\big[1 - y_{i,t}\, u_i^\top x_{i,t}\big]_+ = \sum_{i=1}^{K} \mathbb{1}_{i,t}\, \ell_t(u_i)$$

and $\|\mathbb{1}_t\|_1$ counts the number of mistaken tasks at time t. Solving for $\mu = \sum_{t\in\mathcal{M}} \|\mathbb{1}_t\|_1$ gives

$$\mu \le \sum_{t\in\mathcal{M}} \ell_t^{\mathbb{1}}(U) + \|U\|_* \sqrt{2\sum_{s=1}^{m} D(V_s\|V_{s-1})}\,. \tag{18}$$

Equation (18) is our general starting point for analyzing potential-based matrix multitask algorithms. In particular, the analysis reduces to bounding from above the Bregman term for the specific matrix norm under consideration.

6.2 Specialization to Schatten p-norms

In this section we focus on Schatten p-norms, therefore measuring similarity (or dissimilarity) in terms of spectral properties. This amounts to saying that a set of reference vectors are similar if they span a low dimensional subspace. Along the same lines, we say that a set of K example vectors are dissimilar if their spanned subspace has dimension close to K. The rank of a matrix whose columns are either the reference vectors or the example vectors exactly provides this information. Here we use certain functions of the singular values of a matrix as proxies for its rank. It is easy to see that this leads to a kind of regularization that is precisely enforced through the use of unitarily-invariant norms. In fact, unitarily-invariant matrix norms control the distribution of the singular values of U, thus acting as spectral co-regularizers for the reference vectors—see, for example, Argyriou et al. (2008) for recent developments on this subject. In different terms, by relying only on the singular values (or on the magnitudes of principal components), unitarily invariant norms are a natural way to determine and measure the most informative directions for a given set of vectors.

For these reasons we now specialize the potential-based matrix classifier of Figure 3 to the Schatten $2p$-norm and set $\|V\| = \|V\|_{s_{2p}} = \|\sigma(V)\|_{2p}$, where V is a generic $d \times K$ matrix, and p is a positive *integer* (thus $2p$ is an even number ≥ 2). Note that, in general,

$$\|V\|_{s_{2p}}^2 = \mathrm{TR}\big((V^\top V)^p\big)^{1/p}.$$

We are now ready to state our main result of this section. The proof, along with surrounding comments, can be found in the appendix.

Theorem 9 *The overall number of mistakes μ made by the $2p$-norm matrix multitask Perceptron (with p positive integer) run on finite sequences of examples $(x_{i,1},y_{1,t}),(x_{i,2},,y_{i,2}),\cdots \in \mathbb{R}^{d\times K}\times\{-1,+1\}$, for $i=1,\ldots,K$, satisfies, for all $U\in\mathbb{R}^{d\times K}$,*

$$\mu \le \sum_{t\in\mathcal{M}} \ell_t^{\mathbb{1}}(U) + (2p-1)\left(\mathbf{M}_{s_{2p}}\|U\|_{s_{2q}}\right)^2 + \mathbf{M}_{s_{2p}}\|U\|_{s_{2q}}\sqrt{(2p-1)\sum_{t\in\mathcal{M}}\ell_t^{\mathbb{1}}(U)}$$

where

$$\mathbf{M}_{s_{2p}} = \max_{t\in\mathcal{M}} \frac{\|M_t\|_{s_{2p}}}{\sqrt{\|\mathbb{1}_t\|_1}}$$

and $\|U\|_{s_{2q}}$ is the Schatten $2q$-norm of U, with $2q=\frac{2p}{2p-1}$.

Remark 10 *In order to verify that in certain cases the bound of Theorem 9 provides a significant improvement over the K independent Perceptron baseline, we focus on the linearly separable case; that is, when the sequences $(x_{i,1},y_{i,1}),(x_{i,2},y_{i,2}),\ldots$ are such that there exists a matrix $U\in\mathbb{R}^{d\times K}$ whose columns u_i achieve a margin of at least 1 on each example: $y_{i,t}u_i^\top x_{i,t}\ge 1$ for all $t=1,2,\ldots$ and for all $i=1,\ldots,K$. In this case the bound of Theorem 9 reduces to*

$$\mu \le (2p-1)\left(\mathbf{M}_{s_{2p}}\|U\|_{s_{2q}}\right)^2 . \tag{19}$$

It is easy to see that for $p=q=1$ the $2p$-norm matrix multitask Perceptron decomposes into K independent Perceptrons, which is our baseline. On the other hand, similarly to the vector case, a trace norm/spectral norm bound can be established when the parameter p is properly chosen. Note first that for basic properties of norms $\|U\|_{s_{2q}}\le\|U\|_{s_1}$ and $\|M\|_{s_{2p}}\le r^{1/(2p)}\|M\|_{s_\infty}$, with $r=\min\{d,K\}$. It now suffices to set $p=\lceil\ln r\rceil$ in order to rewrite (19) as

$$\mu \le (2\ln r+1)e\left(\mathbf{M}_{s_\infty}\|U\|_{s_1}\right)^2$$

where U is now penalized with the trace norm and M_t is measured with the spectral norm $\|\cdot\|_{s_\infty}$. If the columns of U span a subspace of dimension $\ll K$, and the matrices of mistaken examples M_t tend to have K nonzero singular values of roughly the same magnitude, then $\|U\|_{s_1}\approx\|U\|_{s_2}$ while $\mathbf{M}_{s_\infty}^2\approx\mathbf{M}_{s_2}^2/K$. Hence this choice of p may lead to a factor K improvement over the bound achieved by the independent Perceptron baseline. See also Remark 11 below. Note that in Theorem 9 (and in the above argument) what matters the most is the quantification in terms of the spectral properties of U via $\|U\|_{s_{2q}}$. The fact that p has to be a positive integer is not a big limitation here, since $2q=\frac{2p}{2p-1}$ can be made arbitrarily close to 1 anyway.

Remark 11 *The bound of Theorem 9 is not in closed form, since the terms $\|\mathbb{1}_t\|_1$ occur in both the left-hand side (via μ) and in the right-hand side (via $\mathbf{M}_{s_{2p}}$). These terms play an essential role to assess the potential advantage of the $2p$-norm matrix multitask Perceptron. In order to illustrate the influence of $\|\mathbb{1}_t\|_1$ on the bound, let us consider the two extreme cases $\|\mathbb{1}_t\|_1=1$ for all $t\in\mathcal{M}$, and $\|\mathbb{1}_t\|_1=K$ for all $t\in\mathcal{M}$. In the former case, the right-hand side of (19) becomes $(2p-1)\|U\|_{s_{2q}}^2$*

(since $\mathbf{M}_{s_{2p}} = 1$*), which is* always *worse than the baseline case* $p = q = 1$*. In the latter case, the bound becomes*

$$(2p-1)\,\frac{\left(\max_{t\in\mathcal{M}} \|M_t\|_{s_{2p}}^2\right)\|U\|_{s_{2q}}^2}{K}$$

which, according to the discussion in the previous remark, opens up the possibility for the factor K *improvement. This is precisely the reason why the spectral regularization is not advantageous in the adversarially chosen task framework described in Section 2. Since the* K *instance-label pairs are the information obtained for each task at any given time step, it appears reasonable that a multitask approach has a chance to improve when such information is abundant, as is the case when* $\|\mathbb{1}_t\|_1 = K$*, and, at the same time, the tasks to be learned are sufficiently similar. For example, in the extreme case when the* K *tasks do actually coincide, it is as if we had to learn a single task, but received* K *independent pieces of information per time step, rather than just one.*

Remark 12 *The* $2p$*-norm matrix multitask Perceptron algorithm updates the primal vector associated with a given task whenever an example for that task is wrongly predicted. Specifically, at time* t *the mistake-driven update rule for the* i*-th task vector is defined as* $v_{i,s} = v_{i,s-1} + y_{i,t}\,x_{i,t}\,\mathbb{1}_{\{\widehat{y}_{i,t}\neq y_{i,t}\}}$*. It is now straightforward to generalize the above update mechanism to the shared loss framework of Dekel et al. (2007), where the sharing is performed via a norm applied to the vector of task losses. Let* $\ell_t(W)$ *be the vector whose entries are the hinge losses incurred by the* K *columns of* W *and pick any vector norm* $\|\cdot\|$*. The goal is to bound the cumulative shared loss* $\sum_t \|\ell_t(W)\|$*. To do so, introduce an additional parameter* $C > 0$ *and write the update as* $v_{i,s} = v_{i,s-1} + y_{i,t}\,x_{i,t}\,\tau_{i,t}$*, where the vector* $\boldsymbol{\tau}_t = [\tau_{1,t},\ldots,\tau_{K,t}]^\top$ *is such that* $\boldsymbol{\tau}_t = \arg\max_{\boldsymbol{\tau}:\|\boldsymbol{\tau}\|_*\le C} \boldsymbol{\tau}^\top \ell_t(W_{s-1})$ *and* $\|\cdot\|_*$ *is the dual of* $\|\cdot\|$*. Since each entry of* $\boldsymbol{\tau}_t$ *is dependent on all the* K *hinge losses suffered by the* $2p$*-norm matrix multitask Perceptron algorithm at time* t*, the update now acts so as to favor certain tasks over the others according to the shared loss induced by* $\|\cdot\|$*. By adapting our proof to the analysis given in Dekel et al. (2007), it is not hard to show that*

$$\sum_{t=1}^{T} \|\ell_t(W_{t-1})\| \le \sum_{t=1}^{T} \|\ell_t(U)\| + \frac{\|U\|_{S_{2q}}^2}{2C} + \frac{T\mathbf{M}_{s_{2p}}^2}{2}$$

where, in analogy with our previous definitions,

$$\mathbf{M}_{s_{2p}} = \max_{t=1,\ldots,T} \frac{\|M_t\|_{s_{2p}}}{\sqrt{\|\boldsymbol{\tau}_t\|_*}} \qquad \textit{and} \qquad M_t = \sum_{i=1}^{K} \tau_{i,t}\,y_{i,t}\,x_{i,t}\,e_i^\top\ .$$

Observe that $\mathbf{M}_{s_{2p}}$ *depends on* C *through* $\boldsymbol{\tau}_t$*, thus preventing an easy optimization over* C*. Moreover, since the upper bound depends on dual Schatten* $2p$*-norms, the discussions in Remark 10 and Remark 11 still apply, with the caveat that in order to have* $\mathbf{M}_{s_\infty}^2 \approx \mathbf{M}_{s_2}^2/K$ *it must be* $\tau_{1,t} \approx \cdots \approx \tau_{K,t}$*.*

6.2.1 IMPLEMENTATION IN DUAL FORM

As for the algorithms in previous sections, the $2p$-norm matrix multitask Perceptron algorithm can also be implemented in dual variables. Setting $X_t = [x_{1,t},\ldots,x_{K,t}]$, it suffices to observe that the predictions $\widehat{y}_{i,t} = \mathrm{SGN}\big(w_{i,s-1}^\top x_{i,t}\big)$ of the $2p$-norm matrix Perceptron reduces to computing the sign of the diagonal entries of the matrix $(V_{s-1}^\top V_{s-1})^{p-1} V_{s-1}^\top X_t$—recall the expression for ∇G calculated in the proof of Theorem 9. Since matrix V_s is updated additively, it is clear that both $V_{s-1}^\top V_{s-1}$

and $V_{s-1}^{\top} X_t$ do depend on instance vectors $x_{i,t}$ only through inner products. This allows us to turn our $2p$-norm matrix multitask Perceptron into a kernel-based algorithm, and repeat the analysis given here using a standard RKHS formalism—see Warmuth (2009) for a more general treatment of kernelizable matrix learning algorithms.

7. Experiments

We evaluated our multitask algorithms on several real-world data sets. Since we are more interested in the multitask kernel for sequential learning problems rather than the nature of the underlying classifiers, we restricted the experiments for the adversarially chosen task model to the multitask Perceptron algorithm of Section 3. In particular, we compare the performance of the multitask Perceptron algorithm with parameter $b > 0$ to that of the same algorithm run with $b = 0$, which is our multitask baseline. Recall from Subsection 3.2 that $b = 0$ amounts to running an independent standard Perceptron on each task. We also evaluated the $2p$-norm matrix multitask Perceptron algorithm under a similar experimental setting, reporting the achieved performance for different values of the parameter p. Finally, we provide experimental evidence of the effectiveness of the $2p$-norm matrix multitask Perceptron algorithm when applied to a learning problem which requires the simultaneous processing of a significant number of tasks.

In our initial experiments, we empirically evaluated the multitask kernel using a collection of data sets derived from the first 160,000 newswire stories in the Reuters Corpus Volume 1 (RCV1, for details see NIST, 2004). Since RCV1 is a hierarchical multiclass and multilabel data set, we could not use it right away. In fact, in order to evaluate the performance of our multitask algorithms in the presence of increasing levels of correlation among target tasks, we derived from RCV1 a collection of data sets where each example is associated with one task among a set of predefined tasks. We generated eight multitask data sets (D1 through D8) in such a way that tasks in different data sets have different levels of correlation, from almost uncorrelated (D1) to completely overlapped (D8). The number of tasks in each of the eight data sets was set to four.

Roughly speaking, we hand-crafted tasks by clustering categories from the original data set. We started from non intersecting sets of categories, which represent nonrelated tasks, and from there we progressively enlarged the intersection areas, thus obtaining tasks which get closer and closer. The whole process involved several steps. We first defined tasks as sets of RCV1 categories (RCV1 is a multilabel data set where labels are sets of hierarchically organized categories). In order to obtain the four tasks in D1, we first chose four subsets of categories from the initial set of all categories in the RCV1 taxonomy in such a way that each subset is both made up of hierarchically related categories and contains at least 15% of positive examples. More precisely, each of the four tasks in D1 is made up of second-level and third-level categories from one of the four main RCV1 sub-trees (CORPORATE/INDUSTRIAL, ECONOMICS, GOVERNMENT/SOCIAL, MARKETS). Since categories in different tasks belong to *different* sub-trees in the RCV1 taxonomy, and each task is composed by categories from the *same* sub-tree, the resulting four tasks in D1 describe very different but consistent topics. Tasks in D2-D8 are generated as follows. First, task one is kept the same in all the eight data sets. As for the other three tasks, we progressively added categories from the first task and dropped some from their own set of categories. We repeated this process seven times. During the first three times (corresponding to data sets D2, D3, and D4) we augmented task two to four with topics from task one; during the last four times (corresponding to data sets D5-D8) we progressively dropped their own initial categories. The whole process is illustrated in Figure 4. As a

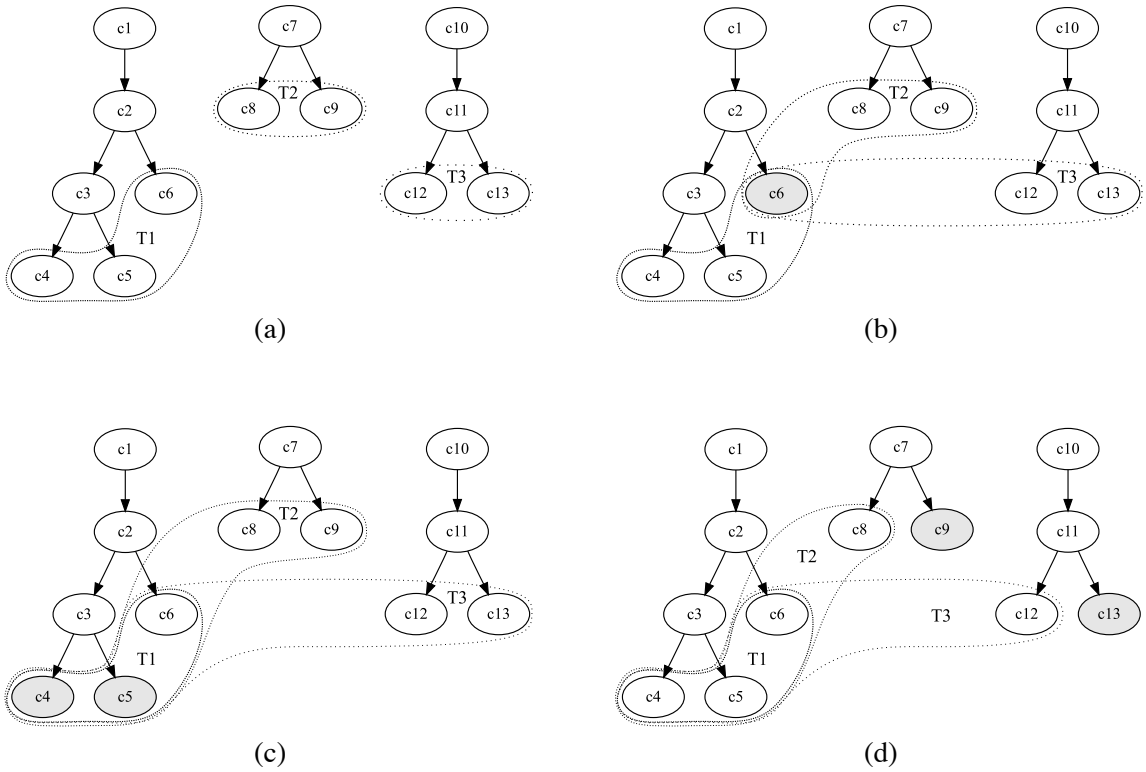


Figure 4: A sample of the task generation process given a taxonomy of 13 categories with three main sub-hierarchies. Tasks are marked as T1, T2 and T3. (a) Uncorrelated tasks are initially defined as sets of categories from different sub-hierarchies. (b and c) Partially overlapped tasks are obtained by first augmenting T2 and T3 with the addition of categories from T1 (category c6 first, then categories c4 and c5), then (d) by shrinking both T2 and T3 with the removal of their initial nodes (these are categories c9 and c13). The shrinking step is repeated until T1, T2 and T3 do coincide.

result of the above construction, as we go from D1 to D8, tasks two, three, and four get both closer to each other and to task one. The last set of four tasks (corresponding to data set D8) is made up of four occurrences of the first task, that is, tasks are completely overlapped in D8.

Once the eight sets of four tasks have been chosen, we generated labels for the corresponding multitask examples as follows. We went through the whole RCV1 data set (whose news example are sorted in chronological order), and gathered examples four by four, where the first example is associated with the first task, the second with the second task, and so on. A multitask example, defined as a set of four (instance, binary label) pairs, was then derived by replacing, for each of the four RCV1 examples, the original RCV1 categories with -1 if the intersection between the associated task and the categories was empty (i.e., if the example did not belong to any of the RCV1 categories which are part of that task), $+1$ otherwise. Since we used 160,000 multilabel and multiclass examples, this process ended up with eight multitask data sets of 40,000 (multitask) examples each.

Subsequences	D1	D2	D3	D4	D5	D6	D7	D8
1-10,000	0.159	0.198	0.215	0.212	0.208	0.203	0.195	0.175
10,001-20,000	0.116	0.158	0.167	0.170	0.167	0.164	0.152	0.134
20,001-30,000	0.104	0.141	0.158	0.155	0.150	0.147	0.138	0.122
30,001-40,000	0.085	0.118	0.125	0.125	0.119	0.113	0.105	0.091

Subsequences	D1	D2	D3	D4	D5	D6	D7	D8
1-10,000	0.395	0.482	0.508	0.499	0.489	0.492	0.461	0.410
10,001-20,000	0.297	0.394	0.428	0.427	0.410	0.394	0.371	0.322
20,001-30,000	0.274	0.374	0.399	0.389	0.375	0.368	0.332	0.291
30,001-40,000	0.231	0.323	0.339	0.337	0.323	0.315	0.287	0.249

Table 1: Online training error rates made by the baseline $b = 0$ on consecutive sequences of multitask examples after a single pass on the data sets D1-D8. The task i_t is chosen either randomly (top) or adversarially (bottom).

We note that the tasks considered here are not linearly separable and, as result of the above construction, different tasks in each data set may have different degrees of nonseparability. This explains why the baseline error rates for data sets D1-D8 are different—see Tables 1 and 2.

Figure 5 shows the fraction of wrongly predicted examples during the online training of the multitask Perceptron algorithm with interaction matrix (5), when the task i_t is chosen either randomly[3] (left) or in an adversarial manner (right). The latter means that i_t is selected so as the resulting signed margin is smallest over the four tasks. This implies that in the first case each task is invoked on average 10,000 times. Nothing can be said in hindsight about the task choices for the adversarial criterion, since this choice is heavily dependent on the online behavior of the classifiers and the noise in the tasks at hand. In both cases we show to what extent the incurred cumulative training error for different values of parameter b exceeds the one achieved by the multitask baseline $b = 0$, depicted here as a straight horizontal line (a negative value means that the chosen value of b achieves an error lower than $b = 0$). Recall that $b = 0$ amounts to running four independent Perceptron algorithms, while $b = 4$ corresponds to running the multitask Perceptron algorithm with the interaction matrix (4). The actual fractions of training error mistakes achieved by the baseline $b = 0$ are reported in Table 1. In order to illustrate how the generalization capabilities of our algorithms progress over time, four pairs of plots are reported, each one showing, from top to bottom, the fraction of mistakes occurred in the example subsequences 1-10,000, 10,001-20,000, 20,001-30,000, and 30,001-40,000, respectively.

Figure 5 confirms that multitask algorithms get more and more competitive as tasks get closer (since tasks are uncorrelated in D1 and totally overlapped in D8). Unsurprisingly, this advantage is higher as we increase the value of b. In fact, Figure 5 clearly shows that the delta errors from $b = 0$ decrease faster, when going from D1 to D8, as we increase b. This amounts to saying that the more we bias our algorithm towards a possible correlation the more we benefit from an actual correlation among tasks. Moreover, it is worth observing that the (rather conservative) choice $b = 1$ obtains an

3. In this case the results are averaged over 3 runs. The observed results were within a 0.002 interval of the plotted values in all 3 runs.

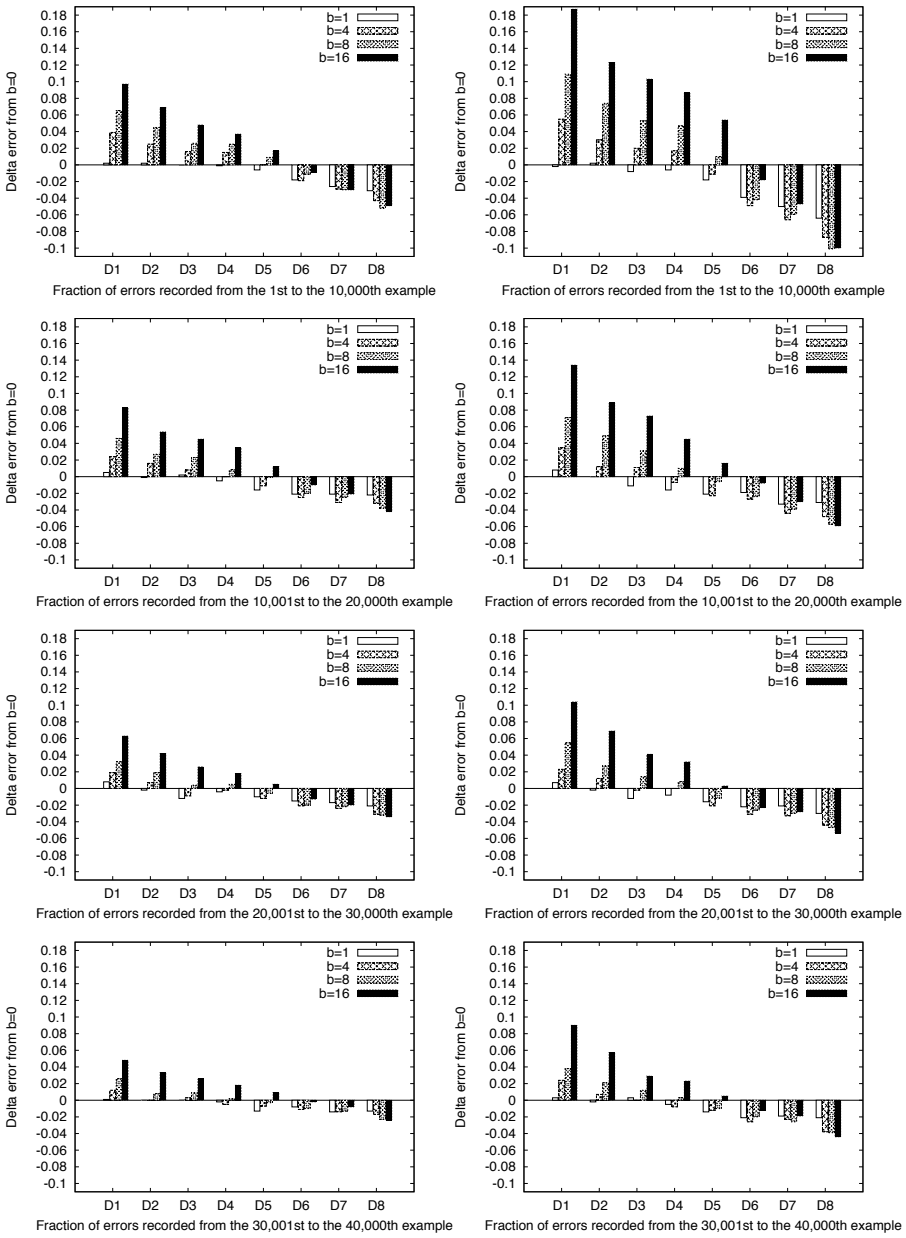


Figure 5: Online behavior of the multitask Perceptron algorithm. We report the extent to which during a single pass over the entire data set the fraction of training mistakes made by the multitask Perceptron algorithm (with $b = 1, 4, 8, 16$) exceeds the one achieved by the baseline $b = 0$, represented here as an horizontal line. On the x-axis are the multitask data sets whose tasks have different levels of correlations, from low (D1) to high (D8). Task indices are randomly chosen in the left plots and adversarially selected in the right ones. The pair of plots on top reports the online training behavior on the first $10,000$ (multitask) examples, the second from top refers to the second $10,000$ examples, and so on.

Subsequences	D1	D2	D3	D4	D5	D6	D7	D8
1-10,000	0.154	0.198	0.207	0.206	0.202	0.196	0.185	0.166
10,001-20,000	0.104	0.141	0.159	0.157	0.152	0.149	0.135	0.120
20,001-30,000	0.095	0.126	0.139	0.138	0.131	0.127	0.115	0.099
30,001-40,000	0.085	0.125	0.132	0.135	0.132	0.124	0.117	0.101

Table 2: Fractions of training errors made by the baseline $p = 1$ on consecutive sequences of multitask examples after a single pass on the data sets D1-D8.

Parameters	Training Error	Test Error	StdDev
P=1	19.0%	13.4%	1.7%
P=2	16.7%	10.9%	1.3%
P=3	16.4%	10.6%	1.6%
P=4	16.6%	9.7%	1.5%
P=5	17.1%	10.2%	1.9%

Table 3: Training errors recorded after a single pass over the spam data set, along with the corresponding test errors. The values are averaged over 40 repetitions of a 10-fold cross-validation scheme. The standard deviation for the test error is reported in parentheses. The standard deviation for the training error is negligible and is therefore omitted.

overall better performance than the multitask baseline $b = 0$, with a higher cumulative error only on the first data set.

We then evaluated the $2p$-norm matrix multitask Perceptron algorithm on the same data set. In this case, we dropped the task choice mechanism since the algorithm is designed to receive all the four instance vectors and to output four predicted labels at each time step. We limited our experiments to the first 10,000 multitask examples. This allows us to make the results achieved by the $2p$-norm matrix multitask Perceptron algorithm somehow comparable (i.t.o. total number of binary labels received) with the ones achieved by the multitask Perceptron algorithm under the random task selection model (the four plots on the left in Figure 5). In Figure 6 we report the (differential) fractions of online training mistakes made by the algorithm on the subsequences 1-2,500, 2,501-5,000, 5,001-7,500 and 7,501-10,000, of multitask examples. The actual fractions of online training error mistakes achieved by the baseline $p = 1$ are showed in Table 2. As expected, the more the tasks get closer to each other the less is the number of wrongly predicted labels output by the $2p$-norm matrix multitask Perceptron algorithm and the larger is the gap from the baseline. In particular, it can be observed that even in D1 the performance of the $2p$-norm matrix multitask Perceptron algorithm is no worse than the one achieved by the baseline. In fact, while our construction tends to guarantee that tasks get closer as we move from D1 to D8 (recall how the tasks are defined in terms of subsets of RCV1 categories), we do not really know in advance how dissimilar the tasks in D1 are, and the performance of the $2p$-norm matrix multitask Perceptron algorithm reveals that they are indeed not so dissimilar.

As a further assessment, we evaluated the empirical performance of the p-norm matrix multitask algorithm on the ECML/PKDD 2006 Discovery Challenge spam data set (for details, see

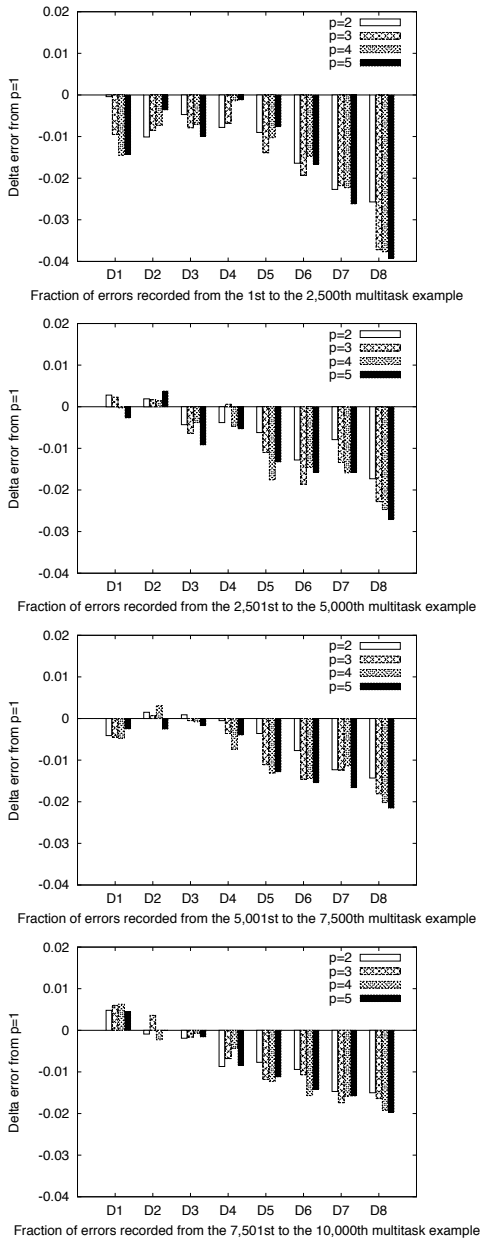


Figure 6: Online behavior of the $2p$-norm matrix multitask Perceptron algorithm. Again, we report the extent to which during a single pass over the entire data set the fraction training mistakes made by the $2p$-norm matrix multitask Perceptron algorithm (with $p = 2,3,4,5$) exceeds the one achieved by the baseline $p = 1$, represented here as an horizontal line. On the x-axis are the multitask data sets whose tasks have different levels of correlations, from low (D1) to high (D8). The top plot reports the online training behavior on the first $2,500$ (multitask) examples, the second from top refers to the second $2,500$ examples, and so on.

ECML/PKDD, 2006). The data set includes 15 sub-data sets, each one containing 400 spam/ham emails for 15 different users. Email messages are encoded by a standard bag-of-words vector representation. Naturally enough, we associated each user with a different task. The experiments were run using a 10-fold cross-validation scheme. Each run consists of a single training epoch followed by a test phase. Since these data are not ordered chronologically as those in RCV1, we repeated the 10-fold cross-validation process 40 times, preceding each repetition with independent shuffles of the 15 data sets. Table 3 shows that the $2p$-norm matrix multitask algorithm exploits underlying latent relations and manages to achieve a significant decrease in both the training and test errors. In particular, the best performance is achieved when p is set to 4, which results in a test error of 9.7%, an improvement of more than 25% relative to the baseline $p = 1$, or nearly a 4.0% decrease in absolute terms. Whereas these are average values, the advantage of the $2p$-norm matrix multitask algorithm is still significant even when the standard deviation is factored in. Moreover, in most runs, the deviation from the average tended to be the on the same side for both $p = 1$ and $p > 1$. In other words, if on a given fold the p-norm matrix multitask Perceptron algorithm with $p > 1$ made a larger number of mistakes than its average, the same held true for the baseline. We stress that the theoretical improvement of a factor K (in this case $K = 15$), is within reach only if the tasks are linearly separable and overlapped. In practice we should and could not expect these conditions to be generally met to their full extent. In particular, while we cannot state how exactly the different spam classification tasks are spectrally related (since the task are not synthetically generated), it is apparent that such relations do actually hold to a certain extent. In fact, by considering the specifics of the learning problem, it is intuitively reasonable that the target spam/ham discriminant functions, though changing from user to user, still share a significant number of common traits.

8. Conclusions and Open Problems

We have studied the problem of sequential multitask learning using two different approaches to formalize the notion of task relatedness: via the Euclidean distance between task vectors, or via a unitarily invariant norm applied to the matrix of task vectors. These two approaches naturally correspond to two different online multitask protocols: one where a single task is selected at each time step, and one where the learner operates simultaneously on all tasks at each time step. We believe that both these protocols have their own merits, each one having its array of possible practical applications. Moreover, while the Schatten-p norm regularization assumption does not make sense in the adversarially chosen task protocol, it is not difficult to adapt the multitask kernel-based algorithm of Section 3 to the fully simultaneous protocol and derive a mistake bound analysis in the lines of the one given in Section 6.

In our worst-case sequential prediction scenario, the best we can hope for i.t.o. prediction performance is a factor K improvement over the baseline running K independent classification algorithms, this is what we essentially achieved in our analysis. We have complemented our theoretical findings with experiments on real-world data sets showing that our algorithms are efficient and can effectively take advantage of latent task relatedness.

We conclude by mentioning a few directions along which our results could be extended.

1. In Section 3.2 it might be interesting to devise methods for dynamically adapting the b parameter as new data are revealed.

2. In Section 3.3 we have shown a simple adaptation to the case when the graph of tasks (i.e., interaction matrix A) is known ahead of time. Is it possible to achieve meaningful bounds when the graph is hidden, and can only be progressively inferred through the choices of i_t?

3. Our multitask regularization techniques rely on the fact that different tasks need to be embedded, either naturally or through some sensible preprocessing, in the same d-dimensional space. It would be interesting to devise a multitask algorithm that does not impose such a constraint.

4. Finally, we believe it would also be interesting to prove *lower* bounds on the number of mistakes as a function of task relatedness.

Acknowledgments

Thanks to Sham Kakade, Massi Pontil, and Francis Bach for useful discussions. We also thank the COLT 2008 reviewers for their comments. This work was supported in part by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the authors' views.

Appendix A.

The following trace inequality, which can be seen as a kind of Holder's inequality applied to non-square matrices, is our main technical tool in the proof of Theorem 9.

Lemma 13 *Let A, B be positive semidefinite matrices, of size $d \times d$ and $K \times K$ respectively, with the same nonzero eigenvalues. Let X be an arbitrary real matrix of size $d \times K$. Then, for any pair on nonnegative exponents $l, g \geq 0$, we have*

$$\mathrm{TR}(X^\top A^l X B^g) \leq \left(\mathrm{TR}(X^\top X)^p\right)^{1/p} \left(\mathrm{TR}\, A^{(l+g)q}\right)^{1/q}$$

where $1/p + 1/q = 1$, $p \geq 1$.

Proof We first consider the case $l \leq g$. By the Cauchy-Schwartz and Holder's inequalities applied to traces (Magnus and Neudecker, 1999, Chapter 11) we have

$$\begin{aligned} \mathrm{TR}(X^\top A^l X B^g) &= \mathrm{TR}\left(B^{(g-l)/2} X^\top A^l X B^{(g+l)/2}\right) \\ &\leq \mathrm{TR}\left(X^\top A^{2l} X B^{g-l}\right)^{1/2} \mathrm{TR}\left(X^\top X B^{g+l}\right)^{1/2} \\ &\leq \mathrm{TR}\left(X^\top A^{2l} X B^{g-l}\right)^{1/2} T_p\left(X^\top X\right)^{1/2} T_q\left(B^{g+l}\right)^{1/2} \end{aligned} \tag{20}$$

where we used the shorthand $T_r(Z) = (\mathrm{TR} Z^r)^{1/r}$. In the case when $l > g$ we can simply swap the matrices $X^\top A^l$ and $X B^g$ and reduce to the previous case.

We now recursively apply the above argument to the left-hand side of (20). Recalling that $T_q(A) = T_q(B)$ and $T_p(X^\top X) = T_p(X X^\top)$, after n steps we obtain

$$\mathrm{TR}\left(X^\top A^l X B^g\right) \leq \left(\mathrm{TR}(X^\top A^{l'} X B^{g'})\right)^{1/2^n} T_p\left(X^\top X\right)^{\sum_{i=1}^n (1/2)^i} T_q\left(B^{g+l}\right)^{\sum_{i=1}^n (1/2)^i}$$

for some pair of exponents $l', g' \geq 0$ such that $l' + g' = l + g$. Since for any such pair l', g', we have $\mathrm{TR}(X^\top A^{l'} X A^{g'}) < \infty$, we can take the limit as $n \to \infty$. Recalling that $\sum_{i=1}^{\infty} (1/2)^i = 1$ completes the proof. ■

We are now ready to prove Theorem 9.

Proof [Theorem 9] We set

$$G(V) = \frac{1}{2} \|V\|_{s_{2p}}^2 = \frac{1}{2} \mathrm{TR}\big((V^\top V)^p\big)^{1/p}.$$

Since $G(V)$ is twice[4] continuously differentiable, by the mean-value theorem we can write

$$D(V_s \| V_{s-1}) = \frac{1}{2} (\mathrm{VEC} M_t)^\top H_G(\xi) \mathrm{VEC}(M_t) \tag{21}$$

where H_G denotes the Hessian matrix of (matrix) function G and ξ is some matrix on the line connecting V_{s-1} to V_s. We start by computing the first derivative,

$$\nabla G(V) = \frac{1}{2} \nabla \mathrm{TR}\big((V^\top V)^p\big)^{1/p} = \frac{1}{2p} \mathrm{TR}\big((V^\top V)^p\big)^{1/p-1} \nabla \mathrm{TR}\big((V^\top V)^p\big) \,. \tag{22}$$

Then, by applying the chain rule to $\nabla\big(\mathrm{TR}(V^\top V)^p\big)$ and using (15) and (16) we obtain

$$\begin{aligned}
\nabla \mathrm{TR}\big((V^\top V)^p\big) &= p \mathrm{VEC}\big((V^\top V)^{p-1}\big)^\top (I_{K^2} + T_{K^2})(I_K \otimes V^\top) \\
&= p \mathrm{VEC}\big((V^\top V)^{p-1}\big)^\top (I_K \otimes V^\top) \\
&\quad + p \Big(T_{K^2} \mathrm{VEC}\big((V^\top V)^{p-1}\big) \Big)^\top (I_K \otimes V^\top) \\
&\quad \text{(since } V^\top V \text{ and } T_{K^2} \text{ are symmetric)} \\
&= 2p \big((I_K \otimes V) \mathrm{VEC}(V^\top V)^{p-1} \big)^\top \\
&= 2p \mathrm{VEC}\big(V (V^\top V)^{p-1} \big)^\top
\end{aligned} \tag{23}$$

the last equation following from (12). We now substitute (23) back into (22) and obtain

$$\nabla G(V) = c(V) \mathrm{VEC}(D)^\top$$

where we set for brevity $D = V B^{p-1}$ and $c(V) = \mathrm{TR}(B^p)^{1/p-1}$ with $B = V^\top V$. Taking the second derivative $H_G = \nabla^2 G$ gives

$$H_G(V) = \mathrm{VEC}(D) \nabla c(V) + c(V) \nabla D \,.$$

Recalling the definition of $c(V)$ and using (15) it is easy to see that $\mathrm{VEC}(D) \nabla c(V)$ is the $Kd \times Kd$ matrix

$$(1-p) \mathrm{TR}\big(B^p\big)^{1/p-2} \mathrm{VEC}(D) \mathrm{VEC}(D)^\top. \tag{24}$$

Since $p \geq 1$ this matrix is negative semidefinite, and we can disregard it when bounding from the above the quadratic form (21). Thus we continue by considering only the second term $c(V) \nabla(D)$ of the Hessian matrix. We have

$$\nabla D = \big(B^{p-1} \otimes I_d\big) + (I_K \otimes V) \nabla B^{p-1}$$

4. In fact G is C^∞ everywhere but (possibly) in zero, since $\mathrm{TR}\big((V^\top V)^p\big)$ is just a polynomial function of the entries of V. Moreover $\mathrm{TR}\big((V^\top V)^p\big) = 0$ if and only if V is the zero matrix.

where

$$\nabla(B^{p-1}) = \left(\sum_{\ell=0}^{p-2} B^\ell \otimes B^{p-2-\ell}\right)(I_{K^2} + T_{K^2})\left(I_K \otimes V^\top\right).$$

Putting together

$$\begin{aligned}(21) \quad \leq \quad & \frac{c(V)}{2}\,\mathrm{VEC}(M_t)^\top (B^{p-1}\otimes I_d)\,\mathrm{VEC}(M_t) \\ & + \frac{c(V)}{2}\mathrm{VEC}(M_t)^\top (I_K \otimes V)\Sigma \times \\ & \times (I_{K^2} + T_{K^2})\left(I_K \otimes V^\top\right)\mathrm{VEC}(M_t) \end{aligned} \tag{25}$$

where we used the shorthand $\Sigma = \sum_{\ell=0}^{p-2} B^\ell \otimes B^{p-2-\ell}$. We now bound the two terms in the right-hand side of (25). Using again (12) combined with (13) we can write

$$\frac{c(V)}{2}\,\mathrm{VEC}(M_t)^\top (B^{p-1}\otimes I_d)\,\mathrm{VEC}(M_t) = \frac{c(V)}{2}\,\mathrm{TR}(M_t^\top X_t B^{p-1}) \leq \frac{1}{2}\,\mathrm{TR}\big((M_t^\top M_t)^p\big)^{1/p}$$

independent of V. The majorization follows from Holder's inequality applied to the positive semidefinite matrices $M_t^\top M_t$ and B^{p-1} (Magnus and Neudecker, 1999, Chapter 11). Moreover, it is easy to see that the symmetric matrices Σ and T_{K^2} commute, thereby sharing the same eigenspace, in fact

$$T_{K^2}\Sigma = \sum_{\ell=0}^{p-2} T_{K^2}\big(B^\ell \otimes B^{p-2-\ell}\big) = \sum_{\ell=0}^{p-2}\big(B^{p-2-\ell}\otimes B^\ell\big)T_{K^2} = \Sigma\, T_{K^2}\ .$$

Hence, $\Sigma\,(I_{K^2} + T_{K^2}) \preccurlyeq 2\Sigma$, and we can bound from above the second term in (25) by

$$c(V)\mathrm{VEC}(M_t)^\top \sum_{\ell=0}^{p-2} B^\ell \otimes A^{p-1-\ell}\mathrm{VEC}(M_t) \tag{26}$$

where we set $A = VV^\top$. Again, (12) and (13) allow us to rewrite this quadratic form as the sum of traces

$$c(V)\sum_{\ell=0}^{p-2}\mathrm{TR}(M_t^\top A^{p-1-\ell} M_t B^\ell)\ .$$

Since A and B have the same nonzero eigenvalues, we can apply Lemma 13 to each term and put together as in (25). After simplifying we get

$$(21) \leq \frac{1}{2}\,(2p-1)\mathrm{TR}\big((M_t^\top M_t)^p\big)^{1/p} = \frac{1}{2}\,(2p-1)||M_t||^2_{s_{2p}}.$$

Substituting back into (18), and recalling that $\|U\|_* = \|U\|_{s_{2q}}$ in the case of Schatten norms, yields

$$\begin{aligned}\mu \quad \leq \quad & \sum_{t\in\mathcal{M}} \ell_t^{\mathbb{1}}(U) + \|U\|_{s_{2q}} \sqrt{(2p-1)\sum_{t\in\mathcal{M}} \|M_t\|^2_{s_{2p}}} \\ \leq \quad & \sum_{t\in\mathcal{M}} \ell_t^{\mathbb{1}}(U) + \|U\|_{s_{2q}} \sqrt{(2p-1)\max_{t\in\mathcal{M}} \frac{\|M_t\|^2_{s_{2p}}}{\|\mathbb{1}_t\|_1}\,\mu}\ .\end{aligned}$$

Solving the inequality for μ, and overapproximating via $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ we now obtain

$$\mu \leq \sum_{t \in \mathcal{M}} \ell_t^{\mathbb{1}}(U) + (2p-1)\left(\mathbf{M}_{s_{2p}} \|U\|_{s_{2q}}\right)^2 + \mathbf{M}_{s_{2p}} \|U\|_{s_{2q}} \sqrt{(2p-1)\sum_{t \in \mathcal{M}} \ell_t^{\mathbb{1}}(U)}$$

thereby concluding the proof. ■

The core of the above analysis is an upper bound on the second-order term of the Taylor expansion of the Schatten p-norm around V_{s-1}. Our proof of this bound is based on a direct matrix analysis. A more general bound has independently been derived in Juditsky and Nemirovski (2009, Proposition 3.1) and used in Kakade et al. (2009), where the unitarily invariant norms of our analysis are replaced by general convex functions.

References

J. Abernethy, P.L. Bartlett, and A. Rakhlin. Multitask learning with expert advice. In *Proceedings of the 20th Annual Conference on Learning Theory*, pages 484–498. Springer, 2007.

A. Agarwal, A. Rakhlin, and P. Bartlett. Matrix regularization techniques for online multitask learning. Technical Report UCB/EECS-2008-138, EECS Department, University of California, Berkeley, 2008.

R.K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems 19*, pages 41–48. MIT Press, 2007.

A. Argyriou, C.A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems 20*, pages 25–32. Curran Associates, 2008.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operation Research Letters*, 31:167–175, 2003.

U. Brefeld, T. Gaertner, T. Scheffer, and S. Wrobel. Efficient co-regularised least squares regression. In *Proceedings of the 23rd International Conference on Machine Learning*. Omnipress, 2006.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order Perceptron algorithm. *SIAM Journal on Computing*, 43(3):640–668, 2005.

O. Dekel, P.M. Long, and Y. Singer. Online learning of multiple tasks with a shared loss. *Journal of Machine Learning Research*, 8:2233–2264, 2007.

ECML/PKDD Discovery Challenge, 2006. URL: www.ecmlpkdd2006.org/challenge.html.

T. Evgeniou, C. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:614–637, 2005.

Y. Freund and R.E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3): 265–299, 2003.

A.J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(3):173–210, 2001.

M. Herbster and G. Lever. Predicting the labelling of a graph via minimum p-seminorm interpolation. In *Proceedings of the 22nd Annual Conference on Learning Theory*. Omnipress, 2009.

M. Herbster, M. Pontil, and L. Wainer. Online learning over graphs. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 305–312. Omnipress, 2005.

L. Hogben. *Handbook of Linear Algebra*. CRC Press, 2006.

R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

L. Jacob, F. Bach, and J.P. Vert. Clustered multi-task learning. In *Advances in Neural Information Processing Systems 21*, pages 745–752. Curran Associates, 2009.

A. Juditsky and A. Nemirovski. Large deviations of vector-valued martingales in 2-smooth normed spaces. Manuscript, 2009.

S. Kakade and D. Foster. Multi-view regression via canonical correlation analysis. In *Proceedings of the 20th Annual Conference on Learning Theory*, pages 82–96. Springer, 2007.

S. Kakade, S. Shalev-Shwartz, and A. Tewari. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. Manuscript, 2009.

J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45:301–329, 2001.

A.S. Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analsys*, 2(1):173–183, 1995.

N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, University of California at Santa Cruz, 1989.

G. Lugosi, O. Papaspiliopoulos, and G. Stoltz. Online multi-task learning with hard constraints. In *Proceedings of the 22nd Annual Conference on Learning Theory*. Omnipress, 2009.

J.R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley, 1999.

A. Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7:117–139, 2006.

A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Nauka Publishers, 1978.

NIST, 2004. URL: trec.nist.gov/data/reuters/reuters.html.

V. Sindhwani D.R. Rosenberg. An RKHS for multi-view learning and manifold co-regularization. In *Proceedings of the 26th International Conference on Machine Learning*, pages 976–983. Omnipress, 2008.

V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning Workshop on Learning with Multiple Views*, 2005.

K. Tsuda, G. Raetsch, and M.K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

M.K. Warmuth. Winnowing subspaces. In *Proceedings of the 24th International Conference on Machine Learning*, pages 999–1006. Omnipress, 2007.

M.K. Warmuth. Kernelization of matrix updates. Manuscript, 2009.

M.K. Warmuth and D. Kuzmin. Online variance minimization. In *Proceedings of the 19th Annual Conference on Learning Theory*, pages 514–528. Springer, 2006.

Journal of Machine Learning Research 11 (2010) 2935-2972 Submitted 7/09; Revised 7/10; Published 10/10

Tree Decomposition for Large-Scale SVM Problems

Fu Chang FCHANG@IIS.SINICA.EDU.TW
Chien-Yang Guo ASDGUO@IIS.SINICA.EDU.TW
Xiao-Rong Lin ECLIPSEX527@IIS.SINICA.EDU.TW
Chi-Jen Lu CJLU@IIS.SINICA.EDU.TW
Institute of Information Science
Academia Sinica
Taipei, Taiwan

Editor: Alexander Smola

Abstract

To handle problems created by large data sets, we propose a method that uses a decision tree to decompose a given data space and train SVMs on the decomposed regions. Although there are other means of decomposing a data space, we show that the decision tree has several merits for large-scale SVM training. First, it can classify some data points by its own means, thereby reducing the cost of SVM training for the remaining data points. Second, it is efficient in determining the parameter values that maximize the validation accuracy, which helps maintain good test accuracy. Third, the tree decomposition method can derive a generalization error bound for the classifier. For data sets whose size can be handled by current non-linear, or kernel-based, SVM training techniques, the proposed method can speed up the training by a factor of thousands, and still achieve comparable test accuracy.

Keywords: binary tree, generalization error ïbound, margin-based theory, pattern classification, tree decomposition, support vector machine, VC theory

1. Introduction

Support vector machines (SVMs) have proven very effective for solving pattern classification problems (Cortes and Vapnik, 1995; Vapnik, 1995). Because of the growing trend to apply them to various domains of interest, including bioinformatics, computer vision, data mining and knowledge discovery, the size of training data sets continues to grow at a rapid rate. At the same time, there is an ongoing effort to speed up the SVM training. One approach, called the *numerical technique* in this paper, seeks efficient solutions to SVM optimization problems.

Well-known numerical methods for solving dual optimization problems include sequential minimal optimization (SMO) (Platt, 1998) and SVM$^{\text{light}}$ (Joachims, 1998). Both methods break a large problem into a series of small problems in order to reduce the amount of memory required for computation. SMO, in particular, has proven superior to similar methods, such as the projected conjugated gradient "chunking" algorithm (Burges, 1998) and Osuna's algorithm (Osuna et al., 1997). For solving dual problems, there are now many new and faster methods, including LASVM (Bordes et al., 2005), maximum-gain working set selection (Glasmachers and Igle, 2006), SVM$^{\text{perf}}$ (Joachims, 2006), LaRank (Bordes et al., 2007), Pegasos (Shalev-Shwartz et al., 2007), bundle methods (Smola et al., 2007), and LIBLINEAR (Fan et al., 2008).

A new direction that has attracted increasing interest in recent years uses the stochastic gradient descent (SGD) technique to solve large-scale SVM problems. The advantage of SGD is that it implements an online learning process that converges to an optimal solution in one examination of the training samples. The above-mentioned LASVM, LaRank and Pegasos algorithms apply SGD to dual optimization problems. There are also algorithms that apply SGD to primal optimization problems, for example, NORMA (Kivinen et al., 2004) and SGD-QN (Bordes et al., 2009).

In addition to the methods for solving dual or primal problems, a number of approaches for solving large SVMs have been proposed, including core vector machines (Tsang et al., 2005) and OCAS (Franc and Sonnenburg, 2008). Readers may refer to a useful paper by Menon (2009) for more details of the numerical methods.

Another type of approach, called *data-reduction* in this paper, reduces a large training data set to one or several small data sets. If only one reduced set is obtained, we call the method *single-set reduction* (SSR); and if multiple reduced sets are obtained, we call the method *multiple-set reduction* (MSR). In the latter case, SVM training is conducted on each of the reduced sets and all the SVMs are combined into a final classifier.

We review MSR methods first. Perhaps the simplest MSR method is bagging (Breiman, 1996). It employs a number of down-sampled data sets to train SVMs, which jointly classify a test object based on majority vote. The boosting method (Schapire, 1990; Schapire and Singer, 2000) trains SVMs in a sequential manner, and the training of a particular SVM is dependent on the training and performance of previously trained SVMs. The divide-and-combine strategy (Rida et al., 1999) decomposes an input space into possibly overlapping regions, assigns each region a local predictor, and combines the local predictors to derive a global solution to the prediction problem. The Bayesian committee machine (Tresp, 2000) partitions a large data set into smaller ones, and the SVMs trained on the reduced sets jointly define the posteriori probabilities of the classes into which test objects are categorized. The method proposed by Collobert et al. (2002) divides a set of input samples into smaller subsets, assigns each subset a local expert, and forms a loop to re-assign samples to local experts according to how well the experts perform. The cascade SVM method (Graf et al., 2004) also splits a large data set into smaller sets and extracts support vectors (SVs) from each of them. The resulting SVs are combined and filtered in a cascade of SVMs. A few passes through the cascade ensures that the optimal solution is found.

On the SSR side of the data-reduction approach, the squashing method (Pavlov et al., 2000) uses a likelihood-based squashing technique to obtain a reduced data set, and then trains linear SVMs on that set. The sparse greedy approximation method (Smola and Schölkopf, 2000) constructs a compressed representation of the design matrix involved in the QP problem; while information vector machines (Lawrence et al., 2003) use a sparse Gaussian process to select training samples using criteria based on information-theoretic principles. Clustering-based SVM (Yu et al., 2003) applies a hierarchical clustering algorithm to obtain a reduced data set, which is used to train SVMs. The concept boundary detection (CBD) method (Panda et al., 2006) prepares nearest-neighbor lists as training samples, and uses a special down-sampling technique to extract the data points that lie close to class boundaries. This method can find a single set of near-boundary points for all class pairs. In contrast, many other methods that use SVMs to analyze training samples have to find different reduced sets for different class pairs, since SVMs can only work on one class pair at a time. For more details of data-reduction methods proposed up to 2001, readers may refer to Tresp (2001).

Finally, we remark that the numerical and data reduction approaches, instead of competing, can actually complement each other's functions. The data reduction approach must train SVMs on reduced data sets, so having an efficient numerical method to perform the task would certainly be useful. The numerical approach, on the other hand, could benefit by using an efficient data reduction method to reduce its computational burden.

In this paper, we propose a method that decomposes a large data set into a number of smaller ones and trains SVMs on each of them. This approach can reduce the total training time because the time complexity of training an SVM is in the order of n^2, where n is the number of training samples (Platt, 1998; Joachims, 1998) . If each smaller problem deals with σ samples, then the complexity of solving all the problems is in the order of $(n/\sigma) \times \sigma^2 = n\sigma$, which is much smaller than n^2 if n is significantly higher than σ. Decomposing a large problem into smaller problems has the added benefit of reducing the number of SVs in each of the resultant SVMs. Since a test sample is classified by only one of these SVMs, the decomposition strategy reduces the time required for the testing process in which the number of SVs dominates the complexity of the computation. One additional benefit of the decomposition approach is the ease of using multi-core/parallel/distributed computing for further speedup, since the SVM problems associated with the decomposed regions can be parallelized idealistically.

The proposed method can be categorized as an MSR method. However, it differs from other MSR methods in that it uses a *decision tree* to obtain multiple reduced data sets, whereas other methods use non-supervised clustering (Rida et al., 1999), random sampling (Breiman, 1996), or random partition (Tresp, 2000; Collobert et al., 2002; Graf et al., 2004). We thus call our method a *decision-tree support vector machine* (DTSVM) and the resultant classifier a DTSVM classifier.

A decision tree decomposes a data space recursively into smaller regions. In terms of the ways the regions are formed, a decision tree can be classified into three types: axis-parallel, oblique and Voronoi types. In the axis-parallel type, the regions are bounded by hyperplanes represented as $x_i = c$, where x_i is a feature and c is a real number (Breiman et al., 1984; Quinlan, 1986). In the oblique type, the regions are bounded by hyperplanes represented as $\sum \alpha_i x_i = c$, where α_i are real numbers (Murthy et al., 1994; Bennett and Blue, 1998; Wu et al., 1999; Bennett et al., 2000). In the Voronoi type, the regions are formed as Voronoi cells by way of various clustering techniques (for a survey, see Dattatreya and Kanal, 1985).

In this paper, we take an axis-parallel decision tree as our decomposition scheme because of its speed in both the training and testing phases. The other two types of decision trees can certainly be used as decomposition schemes, but their computational cost is significantly higher than that of the axis-parallel type. Without conducting a tradeoff study, it is rather difficult to determine whether the additional cost would yield a noteworthy benefit; therefore, we have decided not to adopt them at this stage.

A number of studies have attempted to combine decision trees and SVMs. Some of the methods were designed to improve the classification accuracy (e.g., Bennett and Blue, 1998; Wu et al., 1999; Bennett et al., 2000; Ramaswamy, 2006; Tibshirani and Hastie, 2007); while others were designed to speed up the SVM testing process (e.g., Platt et al., 2000; Sahbi and Geman, 2006; Sun et al., 2007). To the best of our knowledge, using a decision tree to speed up the training of multiclass SVMs has not been proposed previously.

Using a decision tree as a decomposition scheme can yield following benefits when dealing with large-scale SVM problems. First, the decision tree may decompose the data space so that certain decomposed regions become homogeneous; that is, they contain samples of the same labels. Then,

in the testing phase, when a data point flows to a homogeneous region, we simply classify it in terms of the common label of that region. This alleviates the burden of SVM training, which is only conducted in heterogeneous regions. In fact, our experiments revealed that, for certain data sets, more than 90% of the training samples reside in homogeneous regions; thus, the decision tree method saves an enormous amount of time when training SVMs. Random partition, on the other hand, cannot produce such an effect, since random pooling of a set of samples can hardly create a homogeneous data set due to the independent sampling operation.

Another benefit of using the decision tree is the convenience it provides when searching for all the relevant parameter values to maximize the solution's validation accuracy, which in turn helps maintain good test accuracy rates. The goal of the DTSVM method is to attain comparable validation accuracy while consuming less time than training SVMs on full data sets. To achieve our objective, we found that it is important to control the size σ of the tree-decomposed regions as well as the SVM-parameter values. For some data sets, σ could be set at 1,500; but for other data sets, it had to be set at a larger value. Thus, in the DTSVM method, σ is an additional parameter to the usual SVM-parameters. Other MSR methods do not attempt to search for the optimal size of decomposed regions. Such searches are particularly easy under the DTSVM method because a decision tree is constructed in a recursive manner; hence, obtaining a tree with a larger size of σ does not require the reconstruction of a decision tree corresponding to that size of σ.

Using a decision tree also reduces the cost of searching for the optimal values of SVM-parameters. Searching for these values is important, but it takes a tremendous amount of time, especially when training non-linear SVMs. To the best of our knowledge, no data-reduction method has attempted to reduce the cost of this operation. Our strategy involves training SVMs with all combinations of SVM-parameter values, but *only* for decomposed regions with an initial σ-level. The optimal values of the SVM-parameters obtained at this level are not necessarily the same as those obtained at higher levels. However, we observe that the best values for a higher level are usually among the top-ranked values for the initial level. Therefore, when we want to train SVMs for a higher σ-level, we only train them with the top-ranked values obtained for the initial level. Given the n^2-complexity of SVM training, restricting the full search of SVM-parameter values to regions with the initial σ-level certainly reduces the SVM training time. In fact, our experiments showed that such savings were possible even when the optimal σ-level was higher than that of the full size data set.

Although the decision tree method may not be the only way to achieve the above benefits for large-scale SVM problems, its effect can be understood in theory and a generalization error bound can be derived for the DTSVM classifier. The bound is the sum of two terms: the first term dominates in magnitude and is associated with SVM training; and the second term is associated with tree training. Our experimental results show that the numerical value of the dominant term is as small as, or of the same order of magnitude as, its counterpart in the generalization error bound for SVM training conducted on the whole data set. This finding constitutes indirect evidence of the efficacy of tree decomposition for large-scale SVM problems.

Finally, we remark that it is possible to have multiple decompositions of the same data space with multiple trees. These trees can be obtained by using a randomized, rather than the optimal, split point at each tree node (Dietterich, 2000). By so doing, we train SVMs on all the decomposed regions and classify the test data based on a majority vote strategy. We have actually studied the effect of such multiple decompositions. In terms of test accuracy, multiple decompositions are not as effective as searching for the optimal σ-level of decomposed regions in a single decision tree. In fact, under the latter search scheme, introducing multiple decompositions does not lead to

any significant improvement. Therefore, to avoid unnecessary complications, we only consider the decomposition of a data space by a single tree in this paper.

In the experimental study, we divided each data set into training, validation and test components. We then used the training component to build DTSVM classifiers, the validation component to determine the optimal parameters, and the test component to measure the test accuracy. We adopted two types of SVM training: one-against-one (1A1) (Knerr et al., 1990) and one-against-others (1AO) (Bottou et al., 1994) . Furthermore, we built non-linear SVMs on the data sets. When evaluating the DTSVM method, we found it could train DTSVM classifiers that achieved comparable test accuracy rates to those of SVM classifiers. For seven medium-size data sets, in which the largest number of sample was 494K and the largest number of feature was 62K, DTSVM achieved speedup factors between 4 and 3,691 for 1A1 training, and between 29 and 5,775 for 1AO training. Moreover, DTSVM achieved much higher speedup factors than several data reduction methods and numerical methods. To demonstrate that DTSVM can train classifiers efficiently for larger data sets, we applied it to four large-size data sets in which the largest number of samples was 4.9M and the largest number of features was 16.6M. For all the data sets, DTSVM could complete 1A1 training and 1AO training within 18.25 hours. Note that the training time included the time required to build a decision tree, the time to train SVMs on all the leaves, and the time to search for the optimal parameters.

The remainder of this paper is organized as follows. In Section 2, we describe the DTSVM method. Section 3 details the experimental results. In Section 4, we provide theoretical results for the DTSVM method. Section 5 contains some concluding remarks.

2. The DTSVM Method

In this section, we describe the decision tree that we use as the decomposition scheme, and discuss the training process for the DTSVM method. An implementation of the DTSVM method is available at

http://ocrwks11.iis.sinica.edu.tw/dar/Download/WebPages/DTSVM.htm

2.1 The Decision Tree

For the decomposition scheme, we adopt CART (Breiman et al., 1984) or a binary C4.5 scheme (Quinlan, 1986) that allows two child nodes to grow from each node that is not a leaf. Using a C4.5 scheme that allows multiple child nodes is feasible; however, we do not consider it in this paper, since a binary C4.5 performs the job rather well for us.

To grow a binary tree, we follow a recursive process, whereby each training sample flowing to a node is sent to its left-hand or right-hand child node. At a given node E, a certain feature f_E of the training samples flowing to E is compared with a certain value v_E so that all samples with $f_E < v_E$ are sent to the left-hand child node, and the remaining samples are sent to the right-hand child node. The values of f_E and v_E are determined as follows. First, the *split point* v_f of each feature f is calculated by

$$v_f = \arg\max_v IR(f, v),$$

where

$$IR(f,v) = I(S) - \frac{|S_{f<v}|}{|S|} I(S_{f<v}) - \frac{|S_{f\geq v}|}{|S|} I(S_{f\geq v}),$$

S is the set of all samples flowing to E; $S_{f<v}$ consists of the elements of S with $f < v$; $S_{f\geq v} = S \backslash S_{f<v}$; $|X|$ is the size of any data set X; and $I(X)$ is the impurity of X. The impurity function used in our experiments is the entropy measure, defined as

$$I(\mathrm{S}) = -\sum_{y} p(S_y) \log p(S_y),$$

where $p(S_y)$ is the proportion of S's samples whose label is y. Then,

$$f_E = \arg\max_{f} IR(f, v_f),$$

and v_E is taken as the split point of f_E.

We stop splitting a node E when one of the following conditions is satisfied: (i) the number of samples that flow to E is smaller than a *ceiling size* σ; or (ii) when $IR(f,v) = 0$ for all f and v at E. The value of σ in the first condition is determined in a data-driven fashion, which we describe in Section 2.2. The second condition occurs mainly in the following cases. (a) All the samples that flow to E are homogeneous; or (b) a subset of them is homogeneous and the remaining samples, although carrying different labels, are identical to some members of the homogeneous subset. There are other possible cases for the second condition, but their occurrence is extremely rare. If we want to split E in these cases, we can choose the following split point to minimize the difference between $|S_{f<v}|$ and $|S_{f\geq v}|$, that is,

$$v_f = \arg\min_{v} \left| |S_{f<v}| - |S_{f\geq v}| \right|.$$

After growing a tree, we train an SVM on each of its leaves, using samples that flow to each leaf as training data (Figure 1). The values of the SVM parameters are also determined in a data-driven fashion. A tree and all SVMs associated with its leaves constitute a DTSVM classifier, as shown in Figure 1. In the training phase, all the SVMs in a DTSVM classifier are trained with the same parameter values. We explain how the optimal values are obtained in Section 2.2. In the validation/testing process, we first input a given validation/test object $\mathbf{x}$ to the tree. If $\mathbf{x}$ reaches a leaf that contains homogeneous samples, we classify $\mathbf{x}$ as the label of those samples; otherwise, we classify it with the SVM associated with that leaf.

2.2 The DTSVM Training Process

Given a training and validation component, we build a DTSVM classifier on the training component and determine its optimal parameter values with the help of the validation component. The parameters associated with a DTSVM classifier are: (i) σ, the ceiling size of the decision tree; and (ii) the SVM parameters. Their optimal values are determined in the following manner.

We begin by training a binary tree with an initial ceiling size σ_0, and then train SVMs on the leaves with SVM-parameters $\theta \in \Theta$, where Θ is the set of all possible SVM-parameter values whose effects we want to evaluate. Note that we express θ in boldface to indicate that it may consist of more than one parameter. Let $v(\sigma_0, \theta)$ be the validation accuracy of the resultant DTSVM classifier.

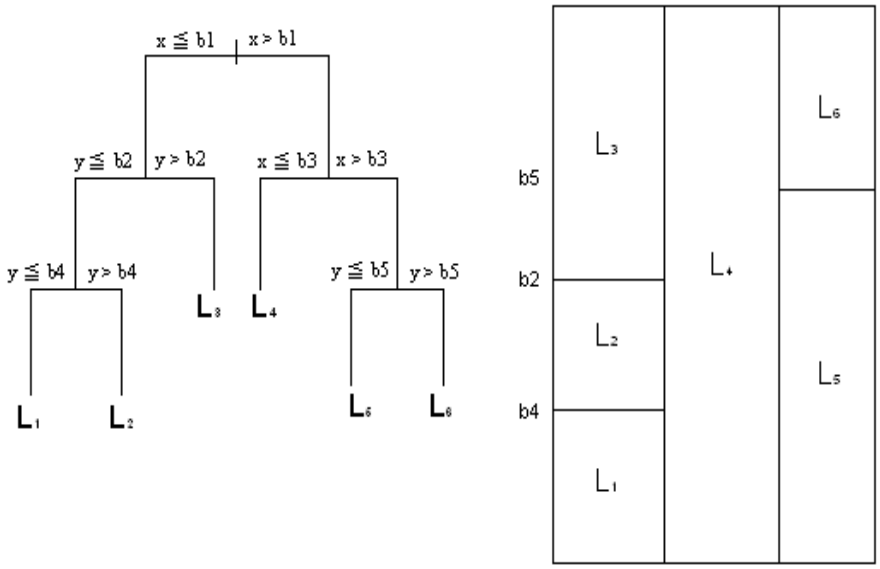


Figure 1: The architecture of a DTSVM classifier: a tree and all its leaves (L1 to L6) are produced and SVMs are trained on the leaves.

Next, we want to construct DTSVM classifiers with larger ceiling sizes, but we only train their associated SVMs with k top-ranked θ. To do this, we rank θ in descending order of $v(\sigma_0, \theta)$. Let Θ_k be the set that consists of k top-ranked θ.

More specifically, we implement the following sub-process, denoted as $SubProcess(\theta)$, for each $\theta \in \Theta_k$.

1. Set $t = 0$ and get the binary tree with the ceiling size σ_0.

2. Increase t by 1 and set $\sigma_t = 4 \times \sigma_{t-1}$. Modify the tree with ceiling size σ_{t-1} to obtain a tree with ceiling size σ_t. This is done by moving from the root towards the leaves and retaining each node whose parent's size is greater than σ_t. Then, train SVMs on the leaves with SVM-parameters θ. Let $v(\sigma_t, \theta)$ be the validation accuracy of the resultant DTSVM classifier.

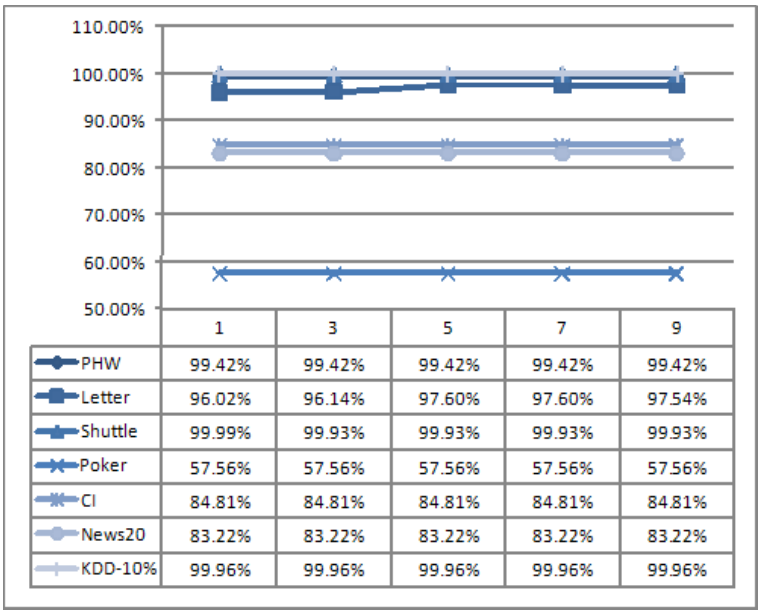
3. If $v(\sigma_t, \theta) - v(\sigma_{t-1}, \theta) \geq 0.5\%$ and σ_t is less than the size of the training component, proceed to step 2.

4. Let $\sigma(\theta) = \sigma_{t-1}$ if $v(\sigma_t, \theta) - v(\sigma_{t-1}, \theta) < 0.5\%$ or $\sigma(\theta) = \sigma_t$ if σ_t is greater than or equal to the size of the training component.

When we have completed $SubProcess(\theta)$ for all $\theta \in \Theta_k$, we define

$$\theta_{opt} = \arg\max_{\theta \in \Theta_k} v(\sigma(\theta), \theta) \text{ and } \sigma_{opt} = \sigma(\theta_{opt}).$$

We then output the DTSVM classifier with the SVM-parameter θ_{opt} and the ceiling size σ_{opt}.



	1	3	5	7	9
PHW	99.42%	99.42%	99.42%	99.42%	99.42%
Letter	96.02%	96.14%	97.60%	97.60%	97.54%
Shuttle	99.99%	99.93%	99.93%	99.93%	99.93%
Poker	57.56%	57.56%	57.56%	57.56%	57.56%
CI	84.81%	84.81%	84.81%	84.81%	84.81%
News20	83.22%	83.22%	83.22%	83.22%	83.22%
KDD-10%	99.96%	99.96%	99.96%	99.96%	99.96%

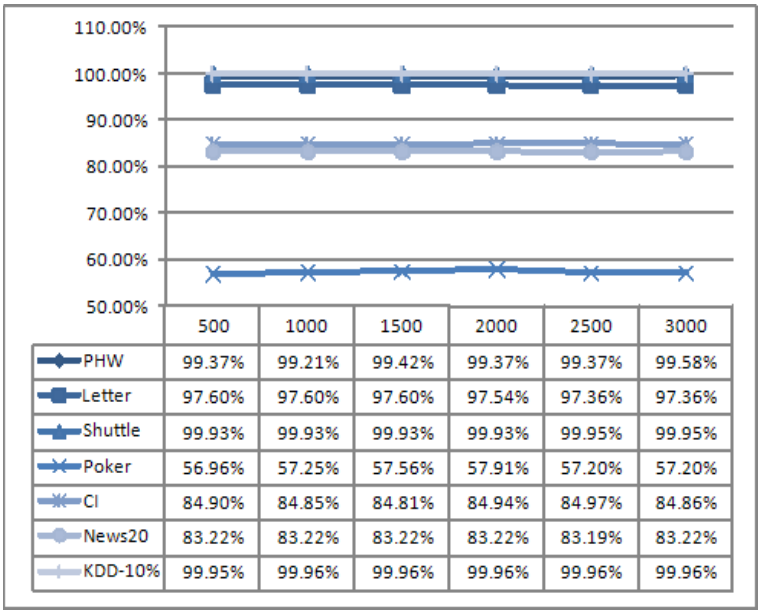
Figure 2: The test accuracy rates obtained by DTSVM on the seven data sets when σ_0 = 1,500 and k = 1, 3, 5, 7 and 9.

Note that, in each *SubProcess*(θ), we set σ_t as quadruple the size (rather than double the size) of σ_{t-1} for two reasons. First, quadrupling the size produces more significant differences between $v(\sigma_t, \theta)$ and $v(\sigma_{t-1}, \theta)$, especially when t is small. This means that if a *SubProcess* terminates at a small t, there is less risk of a low validation accuracy rate. Second, quadrupling the size enables the training process to progress at a faster pace. This means that if a *SubProcess* terminates at a large t, it moves more rapidly towards that end of the process.

The initial ceiling size σ_0 (=1,500) and the number k (=5) of the top-ranked parameters are set heuristically. To observe how these settings impact the test accuracy, we first fix σ_0 at 1,500 and vary k from 1 to 9 at a step size of 2; we then plot the test accuracy rates obtained by DTSVM on the seven data sets whose details are shown in Table 1. Figure 2 shows that any value of k is good for all the data sets except "Letter", while k = 5 or 7 is particularly good for "Letter". Moreover, setting the right value of k improves the test accuracy of "Letter" significantly. Next, we fix k at 5 and vary σ_0 from 500 to 3,000 with a step size of 500. As shown in Figure 3, varying the value of σ_0 does *not* affect the test accuracy of any data set significantly.

3. Experimental Results

In this section, we describe the data sets used in the experiments and the methods that we compared. We then present and discuss the experimental results.



	500	1000	1500	2000	2500	3000
PHW	99.37%	99.21%	99.42%	99.37%	99.37%	99.58%
Letter	97.60%	97.60%	97.60%	97.54%	97.36%	97.36%
Shuttle	99.93%	99.93%	99.93%	99.93%	99.95%	99.95%
Poker	56.96%	57.25%	57.56%	57.91%	57.20%	57.20%
CI	84.90%	84.85%	84.81%	84.94%	84.97%	84.86%
News20	83.22%	83.22%	83.22%	83.22%	83.19%	83.22%
KDD-10%	99.95%	99.96%	99.96%	99.96%	99.96%	99.96%

Figure 3: The test accuracy rates obtained by DTSVM on the seven data sets when $k = 5$, and $\sigma_0 =$ 500, 1,000, 1,500, 2,000, 2,500 and 3,000.

3.1 The Data Sets

In the experiments, we divided the data sets into two groups. The first group was used to evaluate the efficiency of DTSVM and some alternative methods in terms of speeding up SVM training. The second group was used to verify that the DTSVM method could handle much larger data sets, for which most of the alternative methods required an excessive amount of time to complete the training process. The first group comprised seven medium-size data sets, ranging from 10K to 494K in size, as shown in Table 1. Most of the data sets have less than 50 features, but the "News20" has 62,060 features. The second group comprised four large-size data sets, ranging from 240K to 4,898K in size, as shown in the Table 2. The "Webspam" data set is not very large in terms of the number of samples (240K), but the number of features is more than 16M; thus, we consider it a large-size data set. All the data sets were obtained from UPI repository (Newman et al., 1998), with the following two exceptions: "PPI", which was used in a protein-protein interaction study (Tseng et al., 2010), and "Webspam", which was obtained from

http://www.cc.gatech.edu/projects/doi/WebbSpamCorpus.html

Note that the actual "Poker" data set in the repository contains 1 million samples; however, we only used its training component in our experiments.

We randomly divided each data set into six parts of approximately equal size, and used four parts as the training component, one part as the validation component, and the remaining part as the test component. The DTSVM classifiers were trained on the training and validation components, as described in Section 2.2. On completion of the training process, we applied the output DTSVM classifier to the corresponding test data set to obtain the test accuracy rate. All the data sets, divided into training, validation and test components, are available on the following website.

Data set	No. of Labels	No. of Samples	No. of Features
Pen Hand Written (PHW)	10	10,992	16
Letter	26	20,000	16
Shuttle	7	58,000	9
Poker	10	25,010	10
Census Income (CI)	2	45,222	14
News20	20	19,927	62,060
KDD CUP 10% (KDD-10%)	5	494,021	41

Table 1: The medium-size data sets used in our experiments.

Data set	No. of Labels	No. of Samples	No. of Features
Forest	7	581,012	54
PPI	2	1,249,814	14
KDD-full	5	4,898,431	41
Webspam	2	240,000	16,609,143

Table 2: The large-size data sets used in our experiments.

http://ocrwks11.iis.sinica.edu.tw/dar/Download/DataSets/DTSVM/datasets.htm

In each data set, we normalized all the feature values to a real number between 0 and 1. We did this by transforming each value v of feature f into $(v - f_{min})/(f_{max} - f_{min})$, where f_{max} and f_{min} are the maximum and minimum values of f respectively.

We only studied non-linear SVMs in our experiments. Moreover, we used the RBF kernel function to measure the similarity between vectors. As a result, we had two SVM parameters: the penalty factor C, whose values were taken from $\Phi = \{10^a : a = -1, 0, \ldots, 5\}$; and the γ parameter in the RBF function, whose values were taken from $\Psi = \{10^b : b = -4, -3, \ldots, 4\}$. Thus, the set of all SVM parameter values was $\Theta = \Phi \times \Psi$, which comprised 63 pairs of values for (C, γ).

SVM training is implemented under the 1A1 and 1AO approaches. When the 1A1 approach is used, there are $n(n-1)/2$ classifiers, where n is the number of labels. Each classifier assigns one of two possible labels to a given validation/test sample. We use all the classifiers to classify a given validation/test sample $\mathbf{x}$, based on a majority vote. Note that a more efficient technique (Platt et al., 2000) that only requires n classifiers can be used in the validation/testing procedure. However, we adopt Knerr et al.'s (1990) technique, which requires $n(n-1)/2$ classifiers, because we are only interested in the relative, rather than the absolute, performance of the methods compared in our experiments. When the 1AO approach is used, there are n decision functions, each of which is associated with a label. We assign $\mathbf{x}$ the label associated with the decision function that yields the highest functional value.

3.2 Methods Compared

The following methods are compared in our experiments.

CART. CART (Breiman et al., 1984) is similar to the decomposition scheme used in DTSVM, but it differs in terms of the stop and classification criteria. In the training phase, CART stops splitting a node when $IR(f,v) = 0$ for all features f and their values v. In the testing phase, it classifies a test sample $\mathbf{x}$ by the label shared by the majority of samples residing at the leaf to which $\mathbf{x}$ flows. Although CART is not designed for speeding up SVMs, it serves here as a benchmark for DTSVM. If CART performs as well as DTSVM in every respect, then there is no need for DTSVM, since CART runs much faster than DTSVM in both the training and testing phases.

RDSVM. RDSVM (randomized SVM) is an alternative to DTSVM that differs from DTSVM in the way it decomposes a data space. In the training phase, when a size σ is given, DTSVM randomly assigns a training sample to one of d subsets, where d is the smallest integer that is greater than or equal to n/σ and n is the number of training samples. RDSVM uses the same procedure as DTSVM to search for the optimal parameters. In the testing phase, RDSVM randomly assigns a test sample to a subset and classifies it according to the SVM associated with that subset.

Bagging. When implementing bagging (Breiman, 1996), we created a number of SVMs for each $\theta \in \Theta$. Each SVM was trained on 1,500 training samples chosen at random. For each θ, the training was conducted sequentially. We stopped at the first m so that the validation accuracy rate of m SVMs did not exceed that of $m-1$ SVMs by more than 0.5%.

CBD. The training process of CBD (Panda et al., 2006) comprises two steps: finding a reduced set, and training an SVM on that set for each $\theta \in \Theta$. The first part involves finding the k-nearest neighbors of each training sample and deriving the reduced data set via a down-sampling technique. Following Panda et al. (2006), we set k at 100. When searching for the 100 nearest neighbors of each training sample $\mathbf{x}$, we keep the current list of 100 nearest neighbors of $\mathbf{x}$. For another training sample $\mathbf{z}$, let $d(\mathbf{x},\mathbf{z})$ be the distance between $\mathbf{x}$ and $\mathbf{z}$. We need to compare this distance with $d(\mathbf{x},\mathbf{w})$, where $\mathbf{w}$ is on the current list and has the largest distance with $\mathbf{x}$. Since the squared distance is the sum of the squared feature differences, we can speed up the comparison by computing the partial sum of $d^2(\mathbf{x},\mathbf{z})$. When this partial sum exceeds $d^2(\mathbf{x},\mathbf{w})$, we stop the comparison and exclude $\mathbf{z}$ from the current list of $\mathbf{x}$.

LIBSVM. LIBSVM (Fan et al., 2005) is now the most widely used software for training and testing SVMs. We take it as the baseline in our experiments; thus, the speedup factor is 1 by assumption. If a compared method is faster in training than LIBSVM, it has a speedup factor above 1.

LASVM. LASVM (Bordes et al., 2005) is a method that solves a dual optimization problem by way of a stochastic gradient descent method that converges to an optimal solution in one examination of the training samples.

LIBLINEAR. LIBLINEAR (Fan et al., 2008) is a fast version of training and testing *linear* SVMs. Its training speed is comparable to, or even faster than, that of Pegasos (Shalev-Shwartz et al., 2007) and SVM$^{\text{perf}}$ (Joachims, 2006). Since LIBLINEAR is *not* a method for speeding up non-linear SVMs, we only include it in our experiments for large-size data sets, for which bagging, CBD, LIBSVM and LASVM take too long to complete the training process. When we train linear SVMs, the values of the penalty factor C are taken from $\Phi = \{10^a : a = -1, 0, \ldots, 5\}$, which comprises 7 real numbers. Furthermore, we require the discriminant function of the classifier to include a bias term.

Among the above methods, DTSVM, RTSVM, bagging and CBD are data reduction methods, while LIBSVM, LASVM and LIBLIEAR are numerical methods.

3.3 Results on Medium-Size Data Sets

The results of applying seven methods, CART, DTSVM, RDSVM, bagging, CBD, LIBSVM and LASVM, to the seven medium-size data sets are shown in Figures 4-8 for 1A1 training, and in Figures 9-13 for 1AO training. In all SVM training sessions, except for LASVM, we used the LIBSVM software (Fan et al., 2005). We adopted all default options of the software, except the parameter values, which we specified in Section 3.1.

Figure 4 and Figure 9 show the training times of the seven methods. The training time of each method comprises the time required to obtain reduced data sets if it is a data reduction method, the time to train all SVMs and the time to search for optimal parameters; however, the time required to input or output data is *not* included. The computation for all the medium-size data sets was performed on an Intel Xeon CPU 3.2 GHz with a 2GB RAM, while that for all the large-size data sets was performed on a Quad-Core Intel Xeon X5365 3.0GHz CPU and 32GB RAM.

Figure 5 and Figure 10 show the *speedup factors* of all the methods except LIBSVM, where the speedup factor of a method $\mathcal{M}$ is computed as LIBSVM's training time divided by $\mathcal{M}$'s training time.

Figure 6 and Figure 11 show the test accuracy rates of the four compared methods. Note that the DTSVM test accuracy is that of the DTSVM classifier with the ceiling size σ_{opt} and SVM-parameters θ_{opt}. When classifying a test sample with SVMs, the most time-consuming part is computing a decision function, whose complexity can be measured in terms of how many SVs are encountered in the classification. Therefore, we use the "*number of encountered support vectors*" (NESV) as a measure of the time-complexity of the test process. NESV is defined as the number of SVs contained in the decision function used to classify a test sample. When a DTSVM or an RDSVM classifier is used, the NESV is associated with the leaf that the test sample flows to. Thus, in the two cases, NESV is the *average* number of SVs encountered by a test sample.

Figure 7 and Figure 12 list the NESVs of the four methods; while Figure 8 and Figure 13 show the *NESV ratios* of all the methods except LIBSVM and CART, where the NESV ratio of a method $\mathcal{M}$ is computed as LIBSVM's NESV divided by $\mathcal{M}$'s NESV.

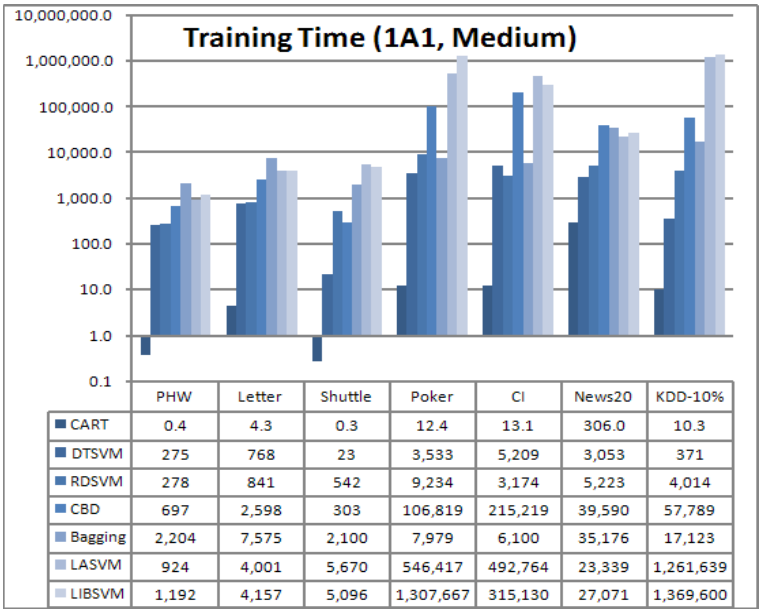
We now summarize the results shown in Figures 4 to 13.

1. There is no doubt that CART was extremely fast in training, but its test accuracy was poor, except on the "Shuttle" and "KDD-10%" data sets, where its accuracy matched the best of all the other methods.

2. In terms of training time, DTSVM outperformed all the other methods, except CART; and in terms of test accuracy and NESV, DSTVM outperformed or performed comparably to all the other methods. It also achieved very large speedup factors and NESV ratios on "Shuttle", "Poker", "CI" and "KDD-10%".

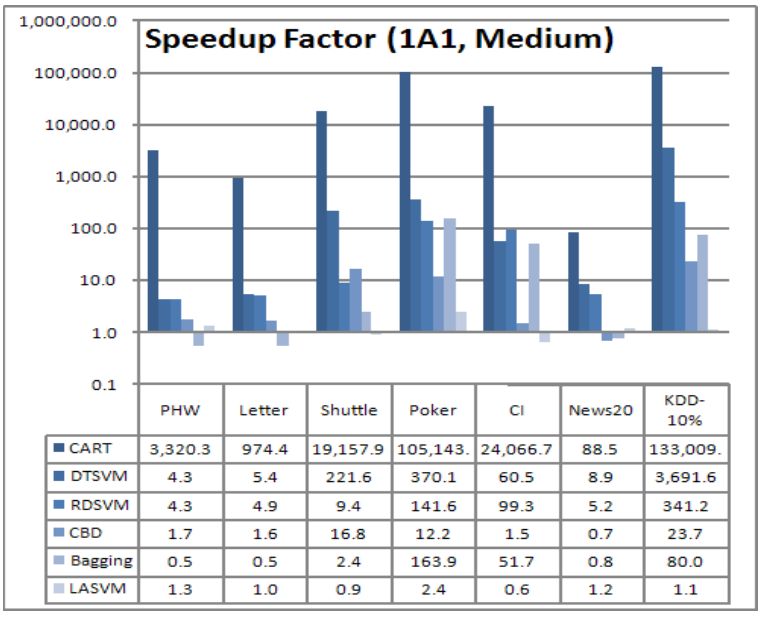
3. RDSVM, being an alternative approach to DTSVM, achieved comparable test accuracy to DTSVM. However, it performed worse in terms of training time on "Shuttle", "Poker" and "KDD-10%". It also performed worse in terms of NESV on "Shuttle", "Poker", "CI" and "KDD-10%".

4. CBD achieved speedup factors above 1 and NESV ratios above 1 on most data sets; however, its scores were overall not as high as those of DTSVM. In addition CBD lagged behind DTSVM in terms of test accuracy on "Letter" and "News20".



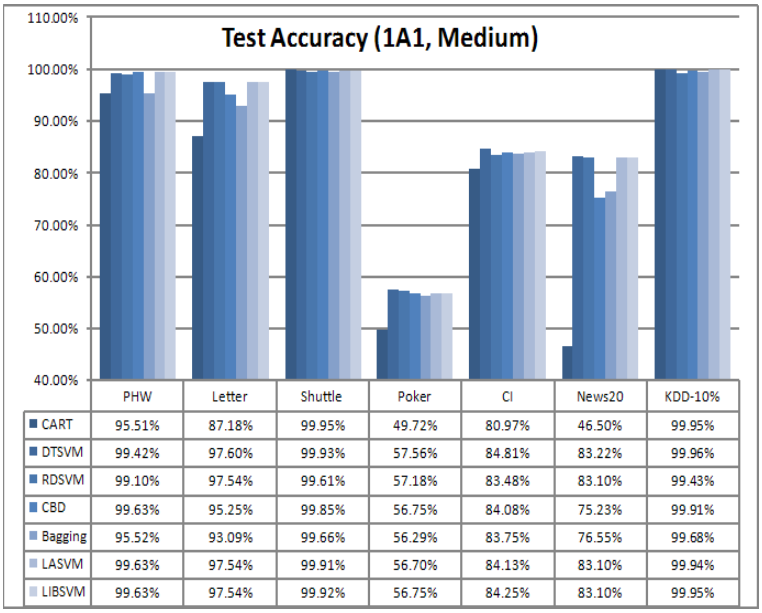
	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
CART	0.4	4.3	0.3	12.4	13.1	306.0	10.3
DTSVM	275	768	23	3,533	5,209	3,053	371
RDSVM	278	841	542	9,234	3,174	5,223	4,014
CBD	697	2,598	303	106,819	215,219	39,590	57,789
Bagging	2,204	7,575	2,100	7,979	6,100	35,176	17,123
LASVM	924	4,001	5,670	546,417	492,764	23,339	1,261,639
LIBSVM	1,192	4,157	5,096	1,307,667	315,130	27,071	1,369,600

Figure 4: Training times of the seven compared methods, expressed in *seconds*. Training type = 1A1. CART, DTSVM and RDSVM outperformed the other methods.



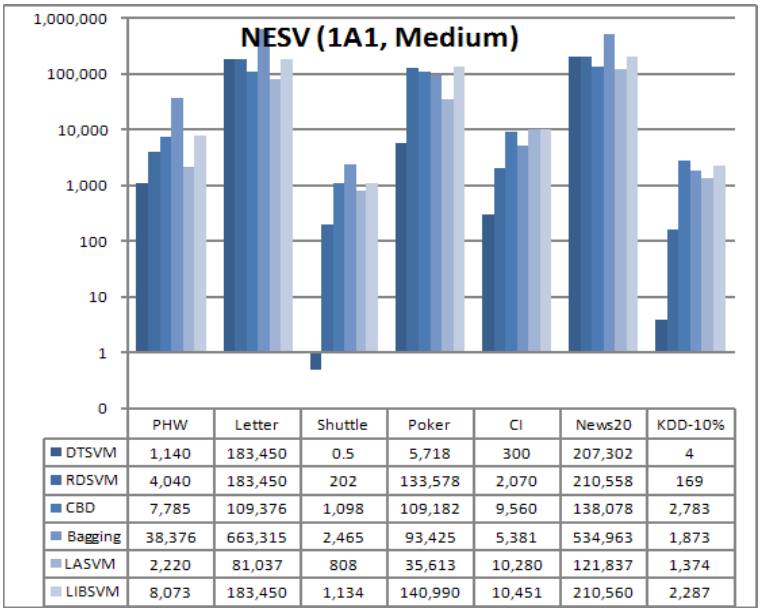
	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
CART	3,320.3	974.4	19,157.9	105,143.	24,066.7	88.5	133,009.
DTSVM	4.3	5.4	221.6	370.1	60.5	8.9	3,691.6
RDSVM	4.3	4.9	9.4	141.6	99.3	5.2	341.2
CBD	1.7	1.6	16.8	12.2	1.5	0.7	23.7
Bagging	0.5	0.5	2.4	163.9	51.7	0.8	80.0
LASVM	1.3	1.0	0.9	2.4	0.6	1.2	1.1

Figure 5: Speedup factors of all the methods except LIBSVM. Training type = 1A1. CART, DTSVM and RDSVM outperformed the other methods.



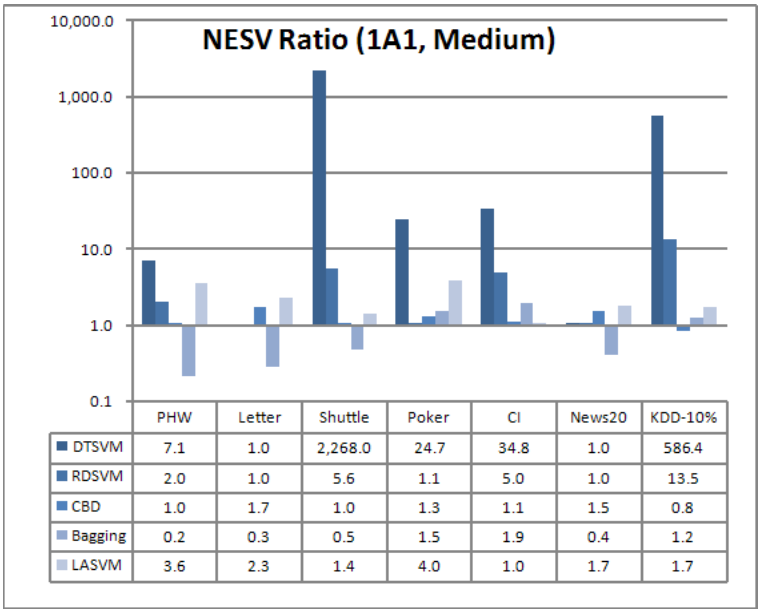
	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
CART	95.51%	87.18%	99.95%	49.72%	80.97%	46.50%	99.95%
DTSVM	99.42%	97.60%	99.93%	57.56%	84.81%	83.22%	99.96%
RDSVM	99.10%	97.54%	99.61%	57.18%	83.48%	83.10%	99.43%
CBD	99.63%	95.25%	99.85%	56.75%	84.08%	75.23%	99.91%
Bagging	95.52%	93.09%	99.66%	56.29%	83.75%	76.55%	99.68%
LASVM	99.63%	97.54%	99.91%	56.70%	84.13%	83.10%	99.94%
LIBSVM	99.63%	97.54%	99.92%	56.75%	84.25%	83.10%	99.95%

Figure 6: Test accuracy rates of all the methods. Training type = 1A1. CART performed poorly on several data sets; while CBD and Bagging lagged behind DTSVM on some data sets.



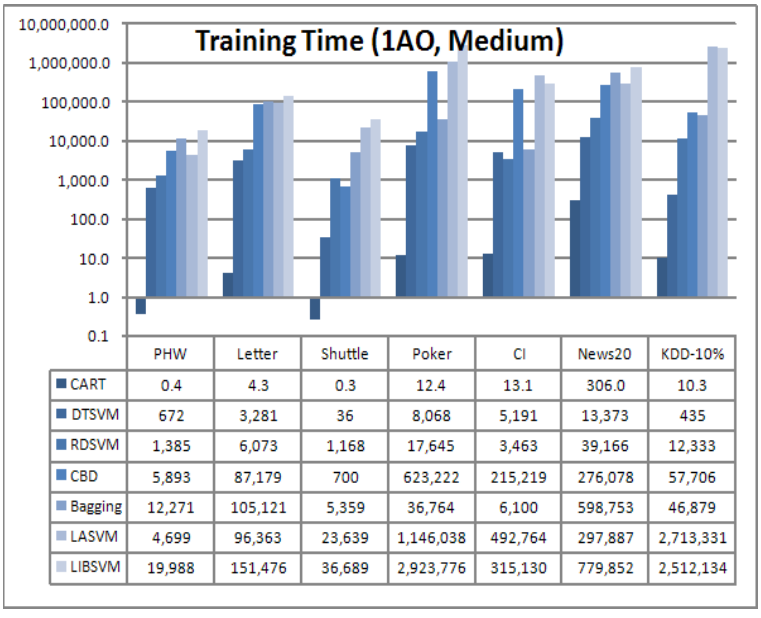
	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
DTSVM	1,140	183,450	0.5	5,718	300	207,302	4
RDSVM	4,040	183,450	202	133,578	2,070	210,558	169
CBD	7,785	109,376	1,098	109,182	9,560	138,078	2,783
Bagging	38,376	663,315	2,465	93,425	5,381	534,963	1,873
LASVM	2,220	81,037	808	35,613	10,280	121,837	1,374
LIBSVM	8,073	183,450	1,134	140,990	10,451	210,560	2,287

Figure 7: The NESVs of all the methods except CART. Training type = 1A1. DTSVM outperformed all the other methods.



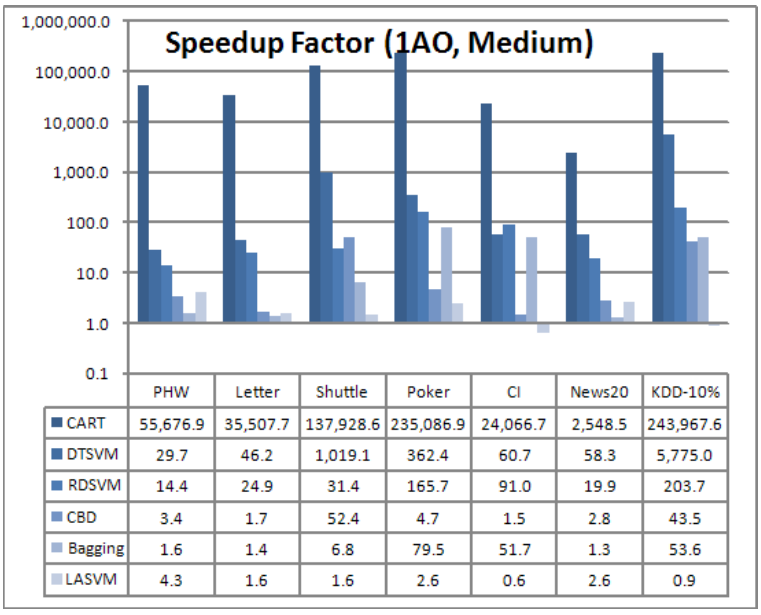
	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
DTSVM	7.1	1.0	2,268.0	24.7	34.8	1.0	586.4
RDSVM	2.0	1.0	5.6	1.1	5.0	1.0	13.5
CBD	1.0	1.7	1.0	1.3	1.1	1.5	0.8
Bagging	0.2	0.3	0.5	1.5	1.9	0.4	1.2
LASVM	3.6	2.3	1.4	4.0	1.0	1.7	1.7

Figure 8: The NESV ratios of all the methods except CART and LISBSM. Training type = 1A1. DTSVM outperformed all the other methods.



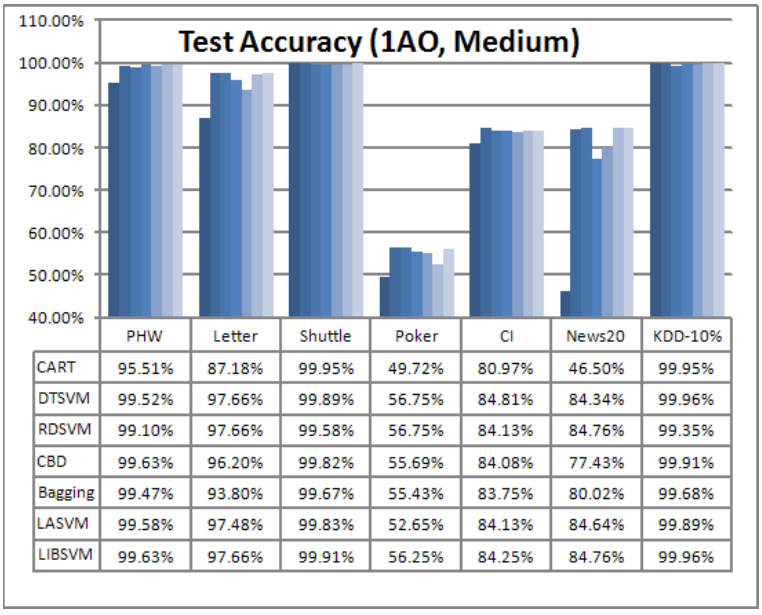
	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
CART	0.4	4.3	0.3	12.4	13.1	306.0	10.3
DTSVM	672	3,281	36	8,068	5,191	13,373	435
RDSVM	1,385	6,073	1,168	17,645	3,463	39,166	12,333
CBD	5,893	87,179	700	623,222	215,219	276,078	57,706
Bagging	12,271	105,121	5,359	36,764	6,100	598,753	46,879
LASVM	4,699	96,363	23,639	1,146,038	492,764	297,887	2,713,331
LIBSVM	19,988	151,476	36,689	2,923,776	315,130	779,852	2,512,134

Figure 9: Training times of all the methods, expressed in *seconds*. Training type = 1AO. CART, DTSVM and RDSVM outperformed the other methods.



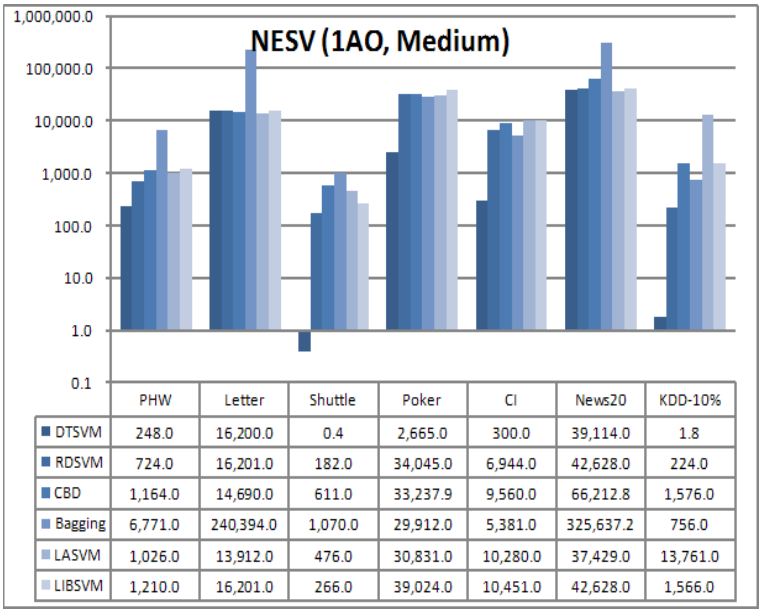
	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
CART	55,676.9	35,507.7	137,928.6	235,086.9	24,066.7	2,548.5	243,967.6
DTSVM	29.7	46.2	1,019.1	362.4	60.7	58.3	5,775.0
RDSVM	14.4	24.9	31.4	165.7	91.0	19.9	203.7
CBD	3.4	1.7	52.4	4.7	1.5	2.8	43.5
Bagging	1.6	1.4	6.8	79.5	51.7	1.3	53.6
LASVM	4.3	1.6	1.6	2.6	0.6	2.6	0.9

Figure 10: Speedup factors of all the methods except LIBSVM. Training type = 1AO. CART, DTSVM and RDSVM outperformed the other methods.



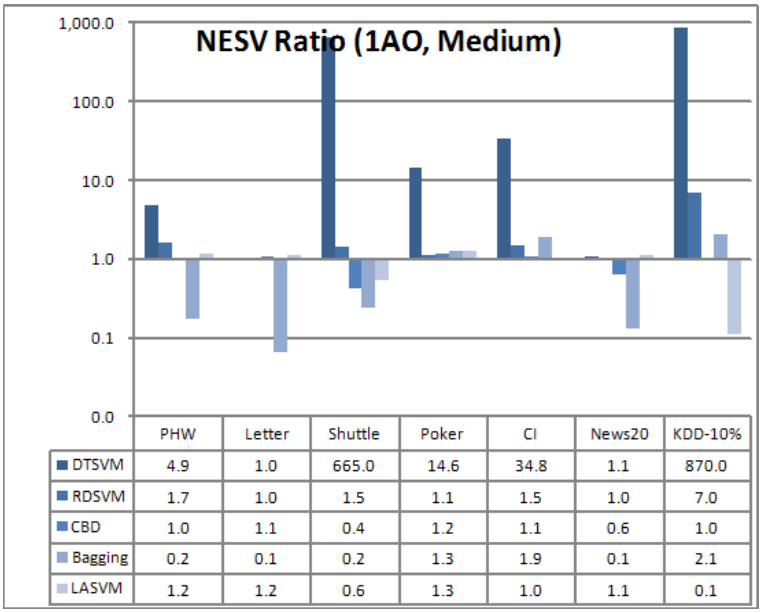
	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
CART	95.51%	87.18%	99.95%	49.72%	80.97%	46.50%	99.95%
DTSVM	99.52%	97.66%	99.89%	56.75%	84.81%	84.34%	99.96%
RDSVM	99.10%	97.66%	99.58%	56.75%	84.13%	84.76%	99.35%
CBD	99.63%	96.20%	99.82%	55.69%	84.08%	77.43%	99.91%
Bagging	99.47%	93.80%	99.67%	55.43%	83.75%	80.02%	99.68%
LASVM	99.58%	97.48%	99.83%	52.65%	84.13%	84.64%	99.89%
LIBSVM	99.63%	97.66%	99.91%	56.25%	84.25%	84.76%	99.96%

Figure 11: Test accuracy rates of all the methods. Training type = 1AO. CART performed poorly on several data sets; while CBD and Bagging lagged behind DTSVM on some data sets.



	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
DTSVM	248.0	16,200.0	0.4	2,665.0	300.0	39,114.0	1.8
RDSVM	724.0	16,201.0	182.0	34,045.0	6,944.0	42,628.0	224.0
CBD	1,164.0	14,690.0	611.0	33,237.9	9,560.0	66,212.8	1,576.0
Bagging	6,771.0	240,394.0	1,070.0	29,912.0	5,381.0	325,637.2	756.0
LASVM	1,026.0	13,912.0	476.0	30,831.0	10,280.0	37,429.0	13,761.0
LIBSVM	1,210.0	16,201.0	266.0	39,024.0	10,451.0	42,628.0	1,566.0

Figure 12: The NESVs of all the methods except CART. Training type = 1AO. DTSVM outperformed all the other methods.



	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
DTSVM	4.9	1.0	665.0	14.6	34.8	1.1	870.0
RDSVM	1.7	1.0	1.5	1.1	1.5	1.0	7.0
CBD	1.0	1.1	0.4	1.2	1.1	0.6	1.0
Bagging	0.2	0.1	0.2	1.3	1.9	0.1	2.1
LASVM	1.2	1.2	0.6	1.3	1.0	1.1	0.1

Figure 13: The NESV ratios of all the methods except CART and LIBSVM. Training type = 1AO. DTSVM outperformed all the other methods.

		PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
1A1	DTSVM	7,694	183,450	181	88,935	8,975	207,290	1,638
	LIBSVM	8,073	183,450	1,134	140,990	10,451	210,560	2,287
1AO	DTSVM	1,775	16,200	195	42,775	8,975	39,122	1,580
	LIBSVM	1,210	16,201	266	39,024	10,451	42,628	1,566

Table 3: The total number of SVs produced by DTSVM and LIBSVM on medium-size data sets. The two methods produced about the same numbers of SVs, even though DTSVM has much smaller NESVs than LIBSVM.

5. Bagging achieved speedup factors below 1 and NESV ratios below 1 on several data sets. It also lagged behind DTSVM in terms of test accuracy on "Letter" and "News20".

6. LASVM, being an alternative numerical method to LIBSVM, achieved speedup factors slightly above 1 on most data sets, but its scores were much lower than those of DTSVM. LASVM also achieved NESV ratios above 1 on most data sets, although they were generally not as high as those of DTSVM. An unexpected result occurred in the 1AO training on "KDD-10%", where LASVM obtained very high NESVs compared to LIBSVM, resulting in a low NESV ratio (0.1). We double checked the process to confirm that the above result was correct.

Finally, we provide some additional information about DTSVM. In Table 3, we show the *total number of support vectors* (TNSV) produced by DTSVM and LIBSVM on all medium-size data sets. TNSV expresses the space-complexity of a training process, whereas NESV expresses the time-complexity of a test process. For DTSVM, the NESV is smaller than the TNSV in most cases, because a test sample usually encounters only some, rather than all, SVs. For LIBSVM, NESV is always the same as TNSV. Note that DTSVM achieved much smaller NESVs on many data sets, but it produced about the same TNSV on all the data sets.

Table 4 shows the DTSVM testing times, as well as the LIBSVM and CART testing times for comparison. As expected from the NESV results, DTSVM's testing time is shorter than that of LIBSVM on all the data sets. We further divide DTSVM's testing time into the time spent on the decision-tree component (DTC) and that spent on local SVMs (lSVMs). In Table 4, the times are separated by a semi-colon. Clearly, the DTC testing time takes an extremely small proportion of DTSVM's testing time. In fact, it is so small that it cannot be measured accurately by the timing mechanism. CART's testing time, on the other hand, is higher than that of the DTC because CART-trees usually grow to deeper levels than DTC-trees.

3.4 Results on Large-Size Data Sets

The results of applying four methods, CART, DTSVM, RDSVM and LIBLINEAR, to the four large-size data sets are shown in Figures 14-16 for 1A1 training, and in Figures 17-19 for 1AO training. Once again, we used the LIBSVM software for all SVM training sessions of DTSVM and RDSVM.

We summarize the results on large-size data sets as follows.

1. Even though DTSVM was not as fast as CART and LIBLINEAR in training, it achieved consistently high test accuracy rates on all the four data sets. It outperformed LIBLINEAR

		PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
CART		0.063	0.125	0.360	0.157	0.204	0.093	2.187
1A1	DTSVM	0.047	3.844	0.000	0.406	0.219	38.141	0.063
		0; 0.047	0; 3.844	0; 0.000	0; 0.406	0; 0.219	0; 38.141	0; 0.063
	LIBSVM	0.265	4.078	0.187	7.406	8.453	39.062	7.344
1AO	DTSVM	0.062	7.344	0.000	1.125	0.219	126	0.063
		0; 0.062	0; 7.344	0; 0.000	0; 1.125	0; 0.219	0; 126	0; 0.063
	LIBSVM	0.328	7.328	0.250	17.313	8.453	138.000	19.875

Table 4: The testing time required by CART, DTSVM and LIBSVM on medium-size data sets. The time required by DTSVM is lower than that required by LIBSVM. The DTC testing time takes a very small proportion of DTSVM's testing time. CART's testing time is higher than that of DTC.

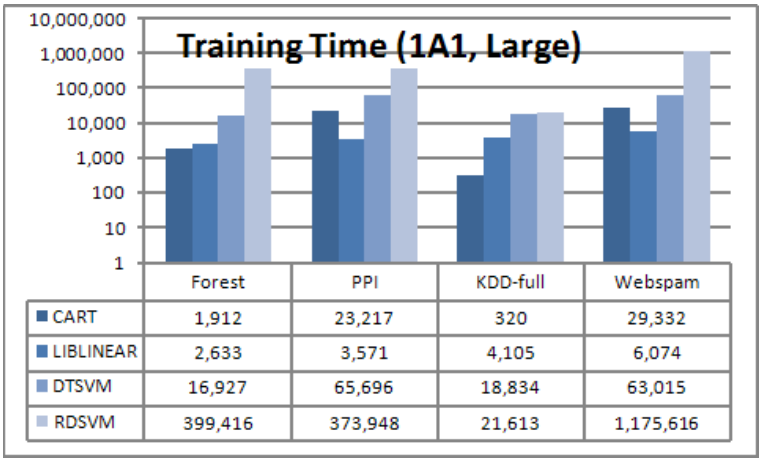
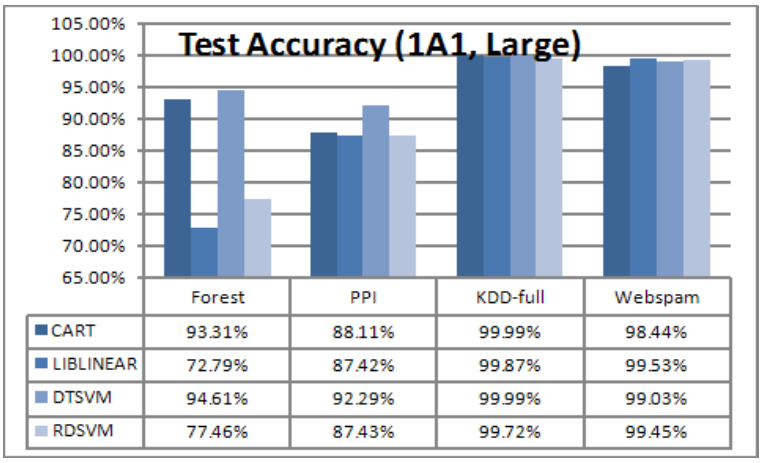


Figure 14: Training times of the four methods, expressed in *seconds*. Training type = 1A1. CART and LIBLINEAR outperformed the other methods.

and RDSVM on "Forest" and "PPI", and surpassed CART on "PPI" by a significant margin. Moreover, DTSVM achieved much lower NESVs than RDSVM on all the data sets.

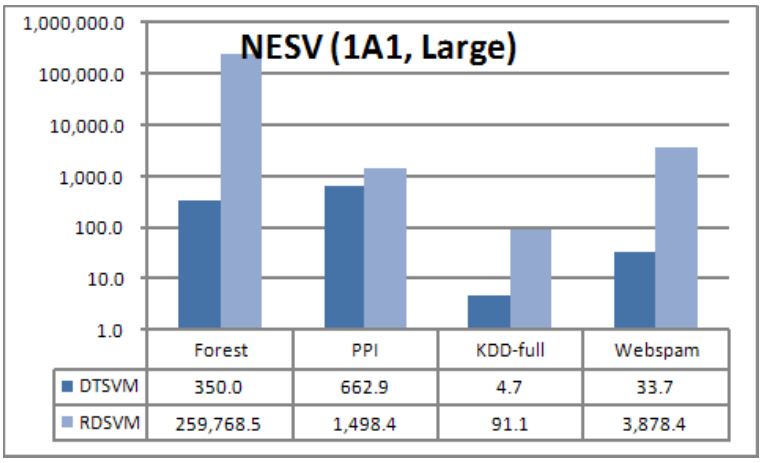
2. Recall that RDSVM achieved equally good test accuracy rates on all the medium-size data sets. However, on the large-size data sets, it achieved much lower accuracy rates on "Forest" and "PPI", and it yielded much higher NESVs on all the data sets. The results show that RDSVM is *not* a good substitute for DTSVM in solving large-scale SVM problems.

3. The results also show that, despite their efficiency in training, CART and LIBLINEAR are *not* good substitutes for DTSVM in solving large-scale problems. LIBLINEAR achieved the best test accuracy rate on "Webspam", presumably because a linear model fits this data set



	Forest	PPI	KDD-full	Webspam
CART	93.31%	88.11%	99.99%	98.44%
LIBLINEAR	72.79%	87.42%	99.87%	99.53%
DTSVM	94.61%	92.29%	99.99%	99.03%
RDSVM	77.46%	87.43%	99.72%	99.45%

Figure 15: Test Accuracy of the four methods. Training type = 1A1. DTSVM outperformed, or performed comparably to, the other methods. CART performed rather well compared to LIBLINEAR.

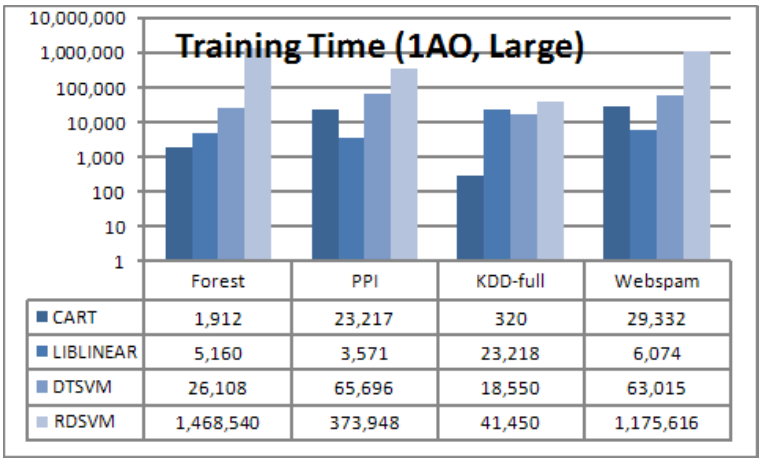


	Forest	PPI	KDD-full	Webspam
DTSVM	350.0	662.9	4.7	33.7
RDSVM	259,768.5	1,498.4	91.1	3,878.4

Figure 16: The NESVs of DTSVM and RDSVM. Training type = 1A1. DTSVM achieved much lower NESVs than RDSVM.

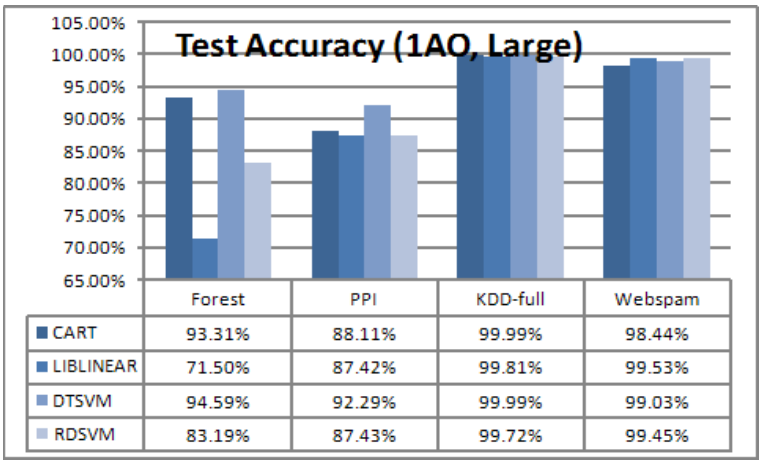
rather well. However, to verify this assumption, we need to compare the test accuracy rates of linear and non-linear models. DTSVM offers us an opportunity to make such a comparison.

We also show the total number of SVs produced by DTSVM in Table 5, while the testing times required by DTSVM and CART are shown in Table 6. DTSVM's testing time is further divided into the amount of time required by the DTC and that required by lSVMs. Once again, it is clear that DTC's testing time only takes a very small proportion of DTSVM's testing time. CART's testing time is higher than that of DTC because CART-trees usually grow to deeper levels than DTC-trees.



	Forest	PPI	KDD-full	Webspam
CART	1,912	23,217	320	29,332
LIBLINEAR	5,160	3,571	23,218	6,074
DTSVM	26,108	65,696	18,550	63,015
RDSVM	1,468,540	373,948	41,450	1,175,616

Figure 17: Training times of the four methods, expressed in *seconds*. Training type = 1AO. CART and LIBLINEAR outperformed the other methods.



	Forest	PPI	KDD-full	Webspam
CART	93.31%	88.11%	99.99%	98.44%
LIBLINEAR	71.50%	87.42%	99.81%	99.53%
DTSVM	94.59%	92.29%	99.99%	99.03%
RDSVM	83.19%	87.43%	99.72%	99.45%

Figure 18: Test Accuracy of the four methods. Training type = 1AO. DTSVM outperformed, or performed as well as, the other methods.

3.5 Further Discussion

To gain insight into why DTSVM is so effective, we show in Table 7 the σ_{opt} derived by DTSVM on medium-size data sets, along with the proportion of training samples that flow to homogeneous leaves. Note that a single table suffices to show all the results because 1A1 training and 1AO training employ the same decision trees and DTSVM yields the same σ_{opt} value for both approaches.

First, we observe that DTSVM required a low ceiling size of 1,500 on all the data sets, except "Letter" and "News20." This explains why DTSVM generally achieved good speedup factors and NESV ratios. Furthermore, the proportion of training samples that flowed to homogeneous leaves under DTSVM was very high in "Shuttle" and "KDD-10%". Since no SVM classifier is involved

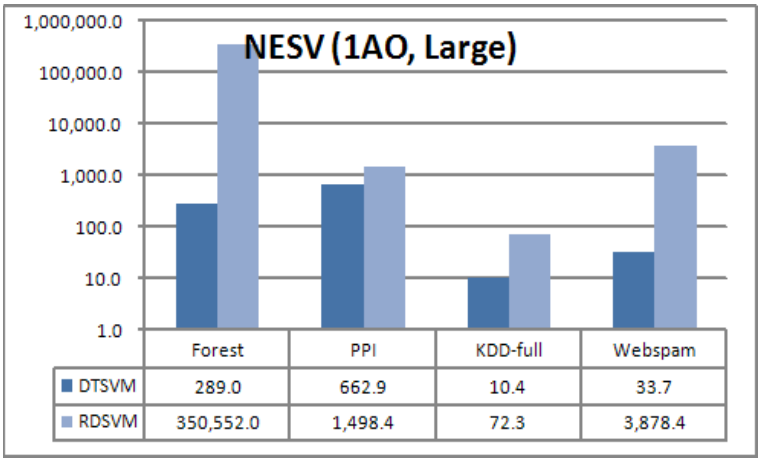


Figure 19: The NESVs of DTSVM and RDSVM. Training type = 1AO. DTSVM achieved much lower NESVs than RDSVM.

		Forest	PPI	KDD-full	Webspam
DTSVM	1A1	140,008	594,687	2,752	8,116
	1AO	114,958	594,687	2,781	8,116

Table 5: The total number of SVs produced by DTSVM on large-size data sets.

		Forest	PPI	KDD-full	Webspam
CART		0.109	0.563	0.281	151.860
DTSVM	1A1	2.485	38.453	1.094	127.000
		0.047; 2.438	0.235; 38.218	0.25; 0.844	75; 52.000
	1AO	3.859	38.453	1.485	127.000
		0.047; 3.812	0.235; 38.218	0.25; 1.235	75; 52.000

Table 6: The testing time required by CART and DTSVM on large-size data sets. DTC's testing time only takes a very small proportion of DTSVM's testing time. CART's testing time is higher than that of DTC.

	PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
σ_{opt}	1,500	24,000	1,500	1,500	1,500	24,000	1,500
Proportion	0%	0%	98.42%	0%	15.76%	0%	97.35%

Table 7: The σ_{opt} obtained by DTSVM on the medium-size data sets and the proportion of training samples that flow to homogeneous leaves.

		PHW	Letter	Shuttle	Poker	CI	News20	KDD-10%
1A1	DTSVM	1,500	24,000	1,500	1,500	1,500	24,000	1,500
	RDSVM	1,500	24,000	1,500	24,000	6,000	24,000	1,500
1AO	DTSVM	1,500	24,000	1,500	1,500	1,500	24,000	1,500
	RDSVM	1,500	24,000	1,500	24,000	24,000	24,000	1,500

Table 8: The σ_{opt} values derived by DTSVM and RDSVM on the medium-size data sets.

Data Set	Training Mode	1,500	6,000	24,000
Letter	1A1	633	45	90
	1AO	2,730	178	373
News20	1A1	1,631	665	757
	1AO	7,056	2,952	3,365

Table 9: The DTSVM training times required for different ceiling sizes.

Data Set	Training Mode	1,500	6,000	24,000
Letter	1A1	95.35%	96.61%	97.60%
	1AO	95.71%	96.91%	97.66%
News20	1A1	67.02%	76.92%	83.22%
	1AO	70.82%	79.06%	84.34%

Table 10: The DTSVM test accuracy rates that correspond to different ceiling sizes.

in any homogeneous leaves, DTSVM achieved very high speedup factors and NESV ratios on these two data sets. The same fact also explains why DTSVM achieved such low NESVs, which even fell below 1 on "Shuttle". Note that this phenomenon occurs because decision trees group neighboring samples into the same leaf. RDSVM, on the other hand, does not produce the same effect because the probability that all samples will carry the same label in the same randomly decomposed region is extremely small.

The lack of homogeneous leaves is not the only reason for RDSVM's poor performance in training. Table 8 shows the σ_{opt} values derived by DTSVM and RDSVM. The results explain why RDSVM achieved much smaller speedup factors and NESV ratios on "CI" and "Poker".

Next, we examine the DTSVM results for the "Letter" and "News20" data sets in which the optimal sizes The σ_{opt} exceeded the size of the training component. Thus, the output DTSVM classifier was trained on the full training component. Even so, DTSVM still achieved speedup factors above 1 because it only trained lSVMs for all the parameter values on leaves with a ceiling size of 1,500, which took much less time than training them on the full training component. The amount of time spent on higher ceiling sizes did not increase at a faster rate, because DTSVM only trained a small number of lSVMs. Moreover, the lSVMs were trained with top-ranked parameters, which tended to require less time than those trained with bottom-ranked parameters.

	Forest	PPI	KDD-full	Webspam
σ_{opt}	1,500	1,500	1,500	1,500
Proportion	5.55%	1.64%	42.05%	80.63%

Table 11: The σ_{opt} values obtained by DTSVM on the large-size data sets and the proportion of training samples that flowed to homogeneous leaves.

		Forest	PPI	KDD-full	Webspam
1A1	DTSVM	1,500	1,500	1,500	1,500
	RDSVM	96,000	1,500	1,500	96,000
1AO	DTSVM	1,500	1,500	1,500	1,500
	RDSVM	1,536,000	1,500	1,500	96,000

Table 12: The σ_{opt} values obtained by DTSVM and RDSVM on the large-size data sets.

Table 9 shows the training times required for different ceiling sizes. We observe that DTSVM spent most of its time on the leaves with the lowest ceiling size. In addition, Table 10 shows the test accuracy rates corresponding to different ceiling sizes, assuming that the training was terminated at those sizes. The results demonstrate the benefit of searching for the σ_{opt} because, if we terminated the training at ceiling size 1,500 or 6,000, we would obtain significantly lower test accuracy rates.

Thus far, we have only discussed the DTSVM results for medium-size data sets. For completeness, Table 11 shows the σ_{opt} values obtained by DTSVM on the large-size data sets, along with the proportion of training samples that flowed to homogeneous leaves. In Table 12, we show the σ_{opt} values obtained by DTSVM and RDSVM. The results explain why RDSVM required much longer training times and more NESVs for "Forest", "PPI" and "Webspam".

Since DTSVM's $\sigma_{opt} = 1,500$ for all four data sets, we do not have any tables for the large-size data sets correspond to Tables 9 and 10 for the medium-size data sets.

4. Generalization Error Bounds for the DTSVM Classifier

To provide a generalization error bound of DTSVM, we start with the following framework. Let $\mathbf{R}^d$ be the d-dimensional Euclidean space. We assume that a set of training examples $X_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ is given, where $(\mathbf{x}_k, y_k) \in \mathbf{R}^d \times \{-1, 1\}$ for $k = 1, \ldots, n$. The DTSVM method produces a classifier $h(\mathbf{x}, \pi, \mathbf{f})$, in which π is a binary tree comprised of L leaves, $\mathbf{f} = (f_1, \ldots, f_L)$, and f_i is related to the lSVM trained on leaf i of π for $i = 1, \ldots, L$. The binary tree π produces a partition function that maps an input in $\mathbf{R}^d$ to $\{1, \ldots, L\}$, and let $\pi(\mathbf{x})$ be the leaf that $\mathbf{x}$ flows to. On the other hand, the lSVM trained on leave i, $i = 1, \ldots, L$ is expressed as $f_i \circ \Phi$, where Φ maps an input in $\mathbf{R}^d$ to a Hilbert space $\mathbf{H}$, and f_i is a *linear* function from $\mathbf{H}$ to $\mathbf{R}$. Note that for a linear function $g : \mathbf{H} \to \mathbf{R}$, there exists some $w \in \mathbf{H}$ such that

$$g(z) = \langle w, z \rangle$$

for all $z \in \mathbf{H}$, and we define $\|g\| = \langle w, w \rangle^{1/2}$.

Note that if $\pi(\mathbf{x}) = i$, then $h(\mathbf{x},\pi,\mathbf{f}) = sign(f_i(\Phi(\mathbf{x}))$. Let us define the function $\mathbf{f}^\pi : \mathbf{R}^d \to \{-1,1\}$ by

$$f^\pi(\mathbf{x}) = f_{\pi(\mathbf{x})}(\Phi(\mathbf{x})).$$

It follows that $h(\mathbf{x},\pi,\mathbf{f}) = sign(\mathbf{f}^\pi(\mathbf{x}))$.

Sometimes, Φ is only defined implicitly. That is, instead of specifying the functional form of Φ, only the inner product of $\Phi(\mathbf{u})$ and $\Phi(\mathbf{v})$ is specified as

$$\langle \Phi(\mathbf{u}), \Phi(\mathbf{v}) \rangle = k(\mathbf{u},\mathbf{v}),$$

where $\mathbf{u},\mathbf{v} \in \mathbf{R}^d$, and $k(\cdot,\cdot)$ is a kernel function. In the remainder of this section, we assume that the function Φ is given and fixed.

Next, we define several notations used in this section. $\mathbf{N}$ is the set of natural numbers; $\mathbf{R}^+$ is the set of positive real numbers; $\mathcal{P}_L(\mathbf{R}^d)$ is the class of all functions from $\mathbf{R}^d$ to $\{1,\ldots,L\}$; $\mathcal{R}(\mathbf{H})$ is the class of all functions from $\mathbf{H}$ to $\mathbf{R}$; and $\mathcal{L}(\mathbf{H})$ is the class of all *linear* functions from $\mathbf{H}$ to $\mathbf{R}$. Moreover, if T is a set, we define $T^L = \{(t_1,\ldots,t_L) : t_i \in T \text{ for } i = 1,\ldots,L\}$; that is, T^L comprises all the L-tuples of T's elements.

To provide a bound for the generalization error of $h(\mathbf{x},\pi,\mathbf{f})$, we need the standard notions of shatter coefficients, margins and covering numbers, which are defined below. Further details can be found in Vapnik (1995) and Cristianini and Shawe-Taylor (2000). First, we define the notion of the shatter coefficient, which we use to measure the complexity of the partition functions corresponding to binary decision trees. Informally, the n^{th} shatter coefficient of a class $\mathcal{G} \subseteq \mathcal{P}_L(\mathbf{R}^d)$ is the maximum number of ways in which n points can be partitioned into L parts by functions in $\mathcal{G}$. Formally, we have the following definition.

Definition 1 *Let $\mathcal{G} \subseteq \mathcal{P}_L(\mathbf{R}^d)$. For any $n \in \mathbf{N}$, the n^{th} shatter coefficient of $\mathcal{G}$ is*

$$V(\mathcal{G},n) = \max_{S \subseteq R^d, |S|=n} |\{\pi_S : \pi \in \mathcal{G}\}|,$$

where π_S is the function obtained by restricting π to the domain S.

Next, we extend the standard notion of the margin to a collection of margins, one for each part of a partition, in the following way.

Definition 2 *Let $\mathbf{f} = (f_1,\ldots,f_L) \in (\mathcal{R}(\mathbf{H}))^L$, $\pi \in \mathcal{P}_L(\mathbf{R}^d)$, $X_n \subseteq \mathbf{R}^d \times \{-1,1\}$, and $\gamma = (\gamma_1,\ldots,\gamma_L) \in (\mathbf{R}^+)^L$. We say that $\mathbf{f}^\pi$ has margin γ on X_n, or $mg(\mathbf{f}^\pi, X_n) \geq \gamma$, if*

$$y \cdot \mathbf{f}^\pi(\mathbf{x}) \equiv y \cdot f_i(\Phi(\mathbf{x})) \geq \gamma_i$$

for any $i \in \{1,\ldots,L\}$ and any $(\mathbf{x},y) \in X_n$ with $\pi(\mathbf{x}) = i$.

In addition, we adopt the following notion of a covering number proposed by Alon et al. (1997). We use it to measure the complexity of the classifiers corresponding to lSVMs. Informally, a covering of a class $\mathcal{F} \subseteq \mathcal{R}(\mathbf{H})$ of functions with respect to a set D of n inputs is a collection of functions in $\mathcal{R}(\mathbf{H})$ such that any function $f \in \mathcal{F}$ is covered by some function g in the collection, in the sense that their values are within some distance η on all the inputs in D. The goal is to find the smallest such collection for any set D of n inputs in some domain E. Formally, we define the covering number as follows.

Definition 3 *Let $\eta \in \mathbf{R}^+$ and $\mathcal{F} \subseteq \mathcal{R}(\mathbf{H})$. For a subset $D \subseteq \mathbf{H}$, let $\mathcal{C}(\mathcal{F}, D, \eta)$ be the smallest collection of functions from D to $\mathbf{R}$ such that, for each $f \in \mathcal{F}$, we have $g \in \mathcal{C}(\mathcal{F}, D, \eta)$ with $|f(z) - g(z)| \leq \eta$ for each $z \in D$. For $E \subseteq \mathbf{H}$ and $n \in \mathbf{N}$, we define the covering number of $\mathcal{F}$ with respect to E, n and η as*

$$N(\mathcal{F}, E, n, \eta) = \max_{D \subseteq E, |D|=n} |\mathcal{C}(\mathcal{F}, D, \eta)|.$$

4.1 Hard Margin Bounds

We have a set X_n of n training examples drawn independently and at random according to the distribution $\mathcal{D}$. Moreover, we have a learned classifier $sign(\mathbf{f}^\pi)$, with $\pi \in \mathcal{P}_L(\mathbf{R}^d)$ and $\mathbf{f} \in (\mathcal{R}(\mathbf{H}))^L$, which classifies all the training examples correctly with a large margin. Our objective is to bound the *generalization error* of the classifier $sign(\mathbf{f}^\pi)$, which is defined as the probability that $sign(\mathbf{f}^\pi(\mathbf{x})) \neq y$, with $(\mathbf{x}, y)$ sampled according to $\mathcal{D}$. The following lemma gives such a bound, which generalizes a known result for SVMs (cf. Cristianini and Shawe-Taylor, 2000).

Lemma 4 *Let $\mathcal{G} \subseteq \mathcal{P}_L(\mathbf{R}^d)$, $\gamma = (\gamma_1, \ldots, \gamma_L) \in (\mathbf{R}^+)^L$, and $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_L$, where $\mathcal{F}_i \subseteq \mathcal{R}(\mathbf{H})$ for $1 \leq i \leq L$. In addition, let $\mathcal{D}$ be a probability distribution on $\mathbf{R}^d \times \{-1, 1\}$, and let n be a large enough integer. Suppose a set of n samples X_n are drawn independently and at random according to $\mathcal{D}$, and consider any classifier $sign(\mathbf{f}^\pi)$, with $\pi \in \mathcal{G}$ and $\mathbf{f} \in \mathcal{F}$, such that $mg(\mathbf{f}^\pi, X_n) \geq \gamma$. Then, with probability $1 - \delta$, the generalization error of $mg(\mathbf{f}^\pi)$ will be at most*

$$\frac{2}{n}\left[\sum_{i=1}^{L} \log N(\mathcal{F}_i, E, 2n, \gamma_i/2) + \log V(\mathcal{G}, 2n) + \log(2/\delta)\right],$$

where $E = \{\Phi(\mathbf{x}) : (\mathbf{x}, y) \in supp(\mathcal{D})\}$ and $supp(\mathcal{D})$ is the support of $\mathcal{D}$.

We provide the proof in Appendix A, as it is rather lengthy and closely follows the standard approach and that of Cristianini and Shawe-Taylor (2000). The idea is to show that the functions $\mathbf{f}^\pi$, with $\pi \in \mathcal{G}$ and $\mathbf{f} \in \mathcal{F}$, can be "well covered" by a small number of functions, so that a union bound can be applied to provide an upper bound on the probability that our classifier has a large generalization error.

Before proceeding further, we explain the meaning of Lemma 4. Suppose for some $\mathcal{G}$ and $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_L$, we can build a classifier $sign(\mathbf{f}^\pi)$ with $\pi \in \mathcal{G}$ and $\mathbf{f} \in \mathcal{F}$ that has a large margin and zero training error. Then, Lemma 4 gives us an upper bound on the generalization error of $sign(\mathbf{f}^\pi)$ in terms of the complexity of $\mathcal{G}$ and $\mathcal{F}_i$, where we measure the complexity of $\mathcal{G}$ by its shatter coefficient $V(\mathcal{G}, 2n)$ and the complexity of each $\mathcal{F}_i$ by its covering number $N(\mathcal{F}_i, E, 2n, \gamma_i/2)$. To obtain a small generalization error, we need to have a $\mathcal{G}$ with a small $V(\mathcal{G}, 2n)$ and an $\mathcal{F}_i$ with a small $N(\mathcal{F}_i, E, 2n, \gamma_i/2)$. However, this does not suggest that we can simply choose any $\mathcal{G}$ and $\mathcal{F}_i$ with small $V(\mathcal{G}, 2n)$ and $N(\mathcal{F}_i, E, 2n, \gamma_i/2)$ for any classification task. This is because we may not be able to build a classifier from $\mathcal{G}$ and $\mathcal{F}_i$ to classify every training example correctly with a large margin. For example, if we do not choose $\mathcal{G}$ properly (say, by using a random partition), the training examples in some parts of the partition may not be separated by SVMs with a large margin. Our main contribution is the discovery that decision trees are good partition functions when combined with SVMs, since they allow a large margin and only require a short training time for lSVMs.

Lemma 4 states a general result for any $\mathcal{G}$ and any $\mathcal{F}_i$ for $1 \leq i \leq L$. We now consider our choice of $\mathcal{G}$ and $\mathcal{F}_i$, which correspond to decision trees and SVMs respectively. For $\beta \in \mathbf{R}^+$, we define

$\mathcal{L}(\mathbf{H},\beta)$ as the class of all linear functions $f \in \mathcal{L}(\mathbf{H})$ with $\|f\| \leq \beta$. A bound can be obtained for the covering number of $\mathcal{L}(\mathbf{H},\beta)$ with respect to E, n and η, provided that E is a bounded subset of $\mathbf{H}$ (see, for example, Bartlett and Shawe-Taylor, 1998). This shows that, although there is an infinite number of functions in $\mathcal{L}(\mathbf{H},\beta)$, they can be covered by a small number of functions in $\mathcal{R}(\mathbf{H})$ with respect to any set of n points in E. As a result, the complexity of the class $\mathcal{L}(\mathbf{H},\beta)$ is low when measured by its covering number.

Lemma 5 *Let $\alpha,\beta,\eta \in \mathbf{R}^+$ and let $n \in \mathbf{N}$. Consider any $E \subseteq \mathbf{H}$ with $\|z\| \leq \rho$ for every $z \in E$. Then, there is a constant c such that*

$$\log N(\mathcal{L}(\mathbf{H},\beta),E,n,\eta) \leq c\frac{\rho^2\beta^2}{\eta^2}\log^2 n.$$

We also define $\mathcal{B}_L(\mathbf{R}^d)$ as the class of partition functions associated with binary trees containing L leaves that partition the space $\mathbf{R}^d$ into L axis-aligned parts, as described in Section 2. Clearly, $\mathcal{B}_L(\mathbf{R}^d) \subseteq \mathcal{P}_L(\mathbf{R}^d)$. The following lemma provides a bound on the n^{th} shatter coefficient of $\mathcal{B}_L(\mathbf{R}^d)$.

Lemma 6 *Let $d,n,L \in \mathbf{N}$. Then*

$$V(\mathcal{B}_L(\mathbf{R}^d),n) \leq L\log(dnL^2).$$

Proof Consider any n-element subset $S \subseteq \mathbf{R}^d$. We need to cut S into L parts. Initially, there is only one part in S. We perform the cut operation iteratively in the following way. Each time, we choose one part from at most $L-1$ existing parts of S and cut it into two parts. To do this, we pick one of the d dimensions and at most one of the $n-1$ cutting hyperplanes on that dimension. Thus, there are at most $(L-1)d(n-1) \leq dnL$ ways to perform one cut operation. To obtain L parts, we repeat the cut operation L-1 times; hence, the number of possible partitions is at most $(dnL)^{L-1} \leq (dnL)^L$. Finally, there are $L!$ ways to order the L parts of each partition, yielding the following result:

$$V(\mathcal{B}_L(\mathbf{R}^d),n) \leq (dnL)^L \cdot (L!) \leq (dnL^2)^L.$$

■

From the above three lemmas, we immediately derive the following theorem.

Theorem 7 *Let $\rho \in \mathbf{R}^+$, $\beta_i \in \mathbf{R}^+$ for $1 \leq i \leq L$, $\gamma = (\gamma_1,\ldots,\gamma_L) \in (\mathbf{R}^+)^L$, and $\mathcal{F} = \mathcal{L}(\mathbf{H},\beta_1) \times \cdots \times \mathcal{L}(\mathbf{H},\beta_L)$. In addition, let $\mathcal{D}$ be a probability distribution on $\mathbf{R}^d \times \{-1,1\}$ such that $\|\Phi(\mathbf{x})\| \leq \rho$ for every $(\mathbf{x},y) \in supp(\mathcal{D})$, and let n be a large enough integer. Suppose a set X_n of n samples are drawn independently and at random according to $\mathcal{D}$, and consider any classifier $sign(\mathbf{f}^\pi)$, with $\pi \in \mathcal{B}_L(\mathbf{R}^d)$ and $\mathbf{f} \in \mathcal{F}$, such that $mg(\mathbf{f}^\pi,X_n) \geq \gamma$. Then, with probability $1-\delta$, the generalization error of $sign(\mathbf{f}^\pi)$ will be at most*

$$\frac{c}{n}\left(\sum_{i=1}^{L}\frac{\rho^2\beta_i^2}{\gamma_i^2}\log^2 n + L\log(dnL^2) + \log(1/\delta)\right),$$

for some constant c.

From Theorem 7, we conclude that if a classifier partitions the training data set into a small number of parts (i.e., L is sufficiently small), and it classifies each training sample with a large margin (i.e., γ_i is large for each i), then the classifier is likely to have a small generalization error. However, this theorem does not indicate how to find a good value for L because, in general, it is hard to know how L affects the margins and the generalization error bound. Instead of being led by any analytical result, the DTSVM learning algorithm takes a data driven approach to find a good L.

Note that the bound in Theorem 7 has a similar form to the generalization error bound of the perceptron decision trees proposed by Bennett et al. (2000). The main difference is that in their bound, γ_i is the margin at the i^{th} internal node and the sum is over all the internal nodes. We remark that it is hard to tell which one of these two bounds is better because, in general, we do not know their actual values for different learning tasks.

4.2 Soft Margin Bounds

Note that Theorem 7 works in the case where the training data X_n can be separated with a margin vector γ. If the data is non-separable or noisy, we need to consider the notion of a soft margin (Cristianini and Shawe-Taylor, 2000). Suppose we have $\pi \in \mathcal{P}_L(\mathbf{R}^d)$, which partitions X_n into L parts: $X_n^1, \ldots, X_n^{(L)}$, where $X_n^{(i)} \equiv \{(\mathbf{x}, y) \in X_n : \pi(\mathbf{x}) = i\}$, and let us denote $X_n^{(i)} = \{(\mathbf{x}_{i,1}, y_{i,1}), \ldots, (\mathbf{x}_{i,n_i}, y_{i,n_i})\}$ for some $n_i \in \mathbf{N}$, for $1 \le i \le L$. In addition, suppose we have $\mathbf{f} = (f_1, \ldots, f_L) \in \mathcal{L}(\mathbf{H}, \beta_1) \times \cdots \times \mathcal{L}(\mathbf{H}, \beta_L)$, where $\beta_i \in \mathbf{R}^+$ for $1 \le i \le L$. Then, we can define the margin slack vector of f_i as follows.

Definition 8 *Let $\gamma = (\gamma_1, \ldots, \gamma_L) \in (\mathbf{R}^+)^L$. For $1 \le i \le L$ and $1 \le j \le n_i$, let*

$$\xi_{i,j} = \max(0, \gamma_i - y_{i,j} \cdot f_i(\Phi(\mathbf{x}_{i,j}))).$$

The definition reflects how far the elements of $X_n^{(i)}$ are from having a margin γ_i. Therefore, for $1 \le i \le L$, we call the vector $\xi_i = (\xi_{i,1}, \ldots, \xi_{i,n_i})$ the margin slack vector of f_i with respect to π and γ_i over X_n.

To find a bound for the generalization error of $sign(\mathbf{f}^{\pi})$ in the case of such a soft margin, we follow the approach of Shawe-Taylor and Cristianini (1999, 2002), which works for the case of $L = 1$. The idea is to map points of $\mathbf{H}$ to a higher dimensional space $\hat{\mathbf{H}}$ so that, for each i, the image of $X_n^{(i)}$ can be separated by some function $\hat{f}_i$ with the desired margin. Theorem 7 can then be applied. To find each $\hat{f}_i$, we present the following lemma, which is a simple extension of Shawe-Taylor & Cristianini's approach. For completeness, we provide the proof in Appendix B.

Lemma 9 *Suppose that $\|\Phi(\mathbf{x})\| \le \rho$ for any $(\mathbf{x}, y) \in supp(\mathcal{D})$. Then, there exists a space $\hat{\mathbf{H}}$, a mapping $\tau_\rho : \mathbf{H} \to \hat{\mathbf{H}}$, and a sequence $\hat{\mathbf{f}} = (\hat{f}_1, \ldots, \hat{f}_L)$ of L functions such that the following four facts hold.*

1. *For any $(\mathbf{x}, y) \in supp(\mathcal{D})$, $\|\tau_\rho(\Phi(x))\| \le \sqrt{2}\rho$.*
2. *For $1 \le i \le L$, $\hat{f}_i \in \mathcal{L}(\hat{\mathbf{H}}, \hat{\beta}_i)$ with $\hat{\beta}_i \le \sqrt{\beta_i^2 + \|\xi_i\|^2/\rho^2}$.*
3. *For $1 \le i \le L$, any $(\mathbf{x}_{i,j}, y_{i,j}) \in X_n^{(i)}$ will be classified correctly with a margin*

$$y_{i,j} \cdot \hat{f}_i(\tau_\rho(\Phi(\mathbf{x}_{i,j}))) \ge \gamma_i.$$

4. For $1 \le i \le L$ *and for any* $(\mathbf{x}, y) \notin X_n$, $\hat{f}_i(\tau_\rho(\Phi(\mathbf{x}))) = f_i(\Phi(\mathbf{x}))$.

According to this lemma, we define

$$\Psi = \tau_\rho \circ \Phi, \text{ and}$$

$$\hat{\mathbf{f}}^\pi(\mathbf{x}) = \hat{f}_{\pi(\mathbf{x})}(\Psi(\mathbf{x})).$$

Then, we know that $\hat{\mathbf{f}}^\pi$ has a margin $(\gamma_1, \ldots, \gamma_L)$ on X_n. Moreover, for $(\mathbf{x}, y) \in \mathit{supp}(\mathcal{D})$, we have

$$\|\Psi(\mathbf{x})\| \le \|\tau_\rho(\Phi(\mathbf{x}))\| \le \sqrt{2}\rho.$$

Now we can apply Theorem 7, with the space $\mathbf{H}$ replaced by $\hat{\mathbf{H}}$ and the mapping Φ replaced by Ψ, to obtain a bound on the generalization error of $\mathit{sign}(\hat{\mathbf{f}}^\pi)$, but with the quantity $\rho^2\beta^2/\gamma^2$ replaced by

$$\frac{2\rho^2(\beta_i^2 + \|\xi_i\|^2/\rho^2)}{\gamma_i^2} = \frac{2(\rho^2\beta_i^2 + \|\xi_i\|^2)}{\gamma_i^2}.$$

Finally, to bound the generalization error of $\mathit{sign}(\mathbf{f}^\pi)$, by Lemma 9, we know that for any $(\mathbf{x}, y) \notin X_n$, $\mathit{sign}(\mathbf{f}^\pi(\mathbf{x})) = \mathit{sign}(\hat{\mathbf{f}}^\pi(\mathbf{x}))$; therefore, $\mathit{sign}(\mathbf{f}^\pi)$ and $\mathit{sign}(\hat{\mathbf{f}}^\pi)$ have the same generalization error for inputs that do not fall within X_n. However, it is possible that the elements of X_n misclassified by $\mathit{sign}(\mathbf{f}^\pi)$ take a nontrivial measure in D, which results in $\mathit{sign}(\mathbf{f}^\pi)$ having a larger generalization error over $\mathcal{D}$ than $\mathit{sign}(\hat{\mathbf{f}}^\pi)$. As suggested by Shawe-Taylor and Cristianini (2002), this can be handled by modifying $\mathit{sign}(\mathbf{f}^\pi)$ on the misclassified elements in X_n. We call this new function the X_n-filtered version of $\mathit{sign}(\mathbf{f}^\pi)$. Then, we have the following theorem.

Theorem 10 *Let* $\rho \in \mathbf{R}^+$, $\beta_i \in \mathbf{R}^+$ *for* $1 \le i \le L$, $\gamma = (\gamma_1, \ldots, \gamma_L) \in (\mathbf{R}^+)^L$, *and* $\mathcal{F} = \mathcal{L}(\mathbf{H}, \beta_1) \times \cdots \times \mathcal{L}(\mathbf{H}, \beta_L)$. *In addition, let* $\mathcal{D}$ *be a probability distribution on* $\mathbf{R}^d \times \{-1, 1\}$ *such that* $\|\Phi(\mathbf{x})\| \le \rho$ *for every* $(\mathbf{x}, y) \in \mathit{supp}(\mathcal{D})$, *and let* n *be a large enough integer. Suppose a set of* n *samples,* X_n, *are drawn independently and at random according to* $\mathcal{D}$; *and consider any* $\pi \in \mathcal{B}(\mathbf{R}^d)$ *and* $\mathbf{f} = (f_1, \ldots, f_L) \in \mathcal{F}$, *such that for* $i \le i \le L$, f_i *has a margin slack vector* ξ_i *with respect to* π *and* γ_i *over* X_n. *Then, with probability* $1 - \delta$, *the generalization error of the* X_n*-filtered version of* $\mathit{sign}(\mathbf{f}^\pi)$ *will be at most*

$$\frac{c}{n}\left(\sum_{i=1}^{L}\left(\frac{\rho^2\beta_i^2 + \|\xi_i\|^2}{\gamma_i^2}\right)\log^2 n + L\log(dnL^2) + \log(1/\delta)\right),$$

for some constant c.

Theorem 10 shows that if we can build a classifier with a small L and a small $\|\xi_i\|$ for every i, then the first two terms inside the parentheses of the bound will be small, and the classifier is likely to have a small generalization error. As with Theorem 7, this does not suggest a way to choose the number L. Instead, the learning algorithm in Section 2 finds a good L in a data-driven manner. To facilitate a better understanding of the bound, we provide numerical values for the two terms derived by DTSVM classifiers. We also compare the numerical values with related quantities derived by global SVM (gSVM) classifiers, that is, SVM classifiers built on the full training data set.

Data Set	1A1		1AO	
	T1	T2	T1	T2
PHW	1,699	55	5,769	55
Shuttle	1,152,495	110	2,756,183	110
CI	202,354	481	202,354	481
Poker	12,253	139	48,893	139
KDD-10%	1,297,678	287	64,238	287

Table 13: The values of two leading terms T_1 and T_2 that appear in the generalization error bound for DTSVM classifiers. Training types = 1A1 and 1AO.

4.3 A Numerical Investigation

Note that in Theorem 10, β_i and γ_i are interdependent quantities for $1 \le i \le L$, so we can fix one of them and try to optimize the other. In the formulation of the SVM optimization problem, the objective is to minimize β_i under the constraint that $\gamma_i = 1$ for $1 \le i \le L$. Under this convention, the bound in Theorem 10 becomes

$$\frac{c}{n}\left(\sum_{i=1}^{L}\left(\rho^2\beta_i^2 + \|\xi_i\|^2\right)\log^2 n + L\log(dnL^2) + \log(1/\delta)\right). \tag{1}$$

Recall that a DTSVM classifier is associated with a tree with L leaves and each leaf is associated with an lSVM. If the tree has only one leaf (i.e., $L = 1$), then the DTSVM classifier will be reduced to a gSVM classifier, which has the following generalization error bound

$$\frac{c}{n}\left(\left(\rho^2\beta^2 + \|\xi\|^2\right)\log^2 n + \log(1/\delta)\right) \tag{2}$$

(cf. Cristianini and Shawe-Taylor, 2000).

Let us compare the terms that appear in parentheses in (1) and (2). The second term $L\log(dnL^2)$ in (1) is the shatter coefficient of the partition function π associated with a binary tree. We claim that the value of (1) achieved by DTSVM is dominated by the first term, which is the sum of L quantities associated with L leaves of a binary tree, as opposed to the single quantity in (2). Moreover, the first term in (1) achieved by DTSVM is comparable to the corresponding quantity in (2) achieved by gSVM. The following results confirm the above two claims.

To validate the first claim, we compare the two leading terms in parentheses in (1). The first term is $T_1 = \sum_{i=1}^{L}(\rho^2\beta_i^2 + \|\xi_i\|^2)\log^2 n$ and the second term is $T_2 = L\log(dnL^2)$. The results, shown in Table 13, confirm the claim that T_1 far exceeds T_2 and (1) is dominated by T_1. Note that we compute T_1 under the following assumptions. (i) When the data set contains more than two labels, T_1 is taken as the average of the quantities over all classifiers. (ii) The value of ρ is always 1 when RBF kernels are involved. (iii) The value of $\|\xi_i\|^2$ is obtained from the solution to the quadratic programming optimization problem. Further details can be found in Cristianini and Shawe-Taylor (2000), Section 6.1.2.

To validate the second claim, we compare $R = \sum_{i=1}^{L}(\rho^2\beta_i^2 + \|\xi_i\|^2)$ and $S = \rho^2\beta^2 + \|\xi\|^2$, which are derived, respectively, from the first terms in the generalization error bounds for DTSVM and

Data Set	1A1			1AO		
	S	R	R/L	S	R	R/L
PHW	74	133	17	326	450	56
Shuttle	114,786	75,630	5,402	593,967	180,990	12,928
CI	16,422	13,599	257	16,422	13,599	257
Poker	370	874	49	2,328	3,486	194
KDD-10%	100,227	70,796	2,212	5,089	3,505	110

Table 14: The values of S and R, which appear in the generalization error bound for gSVM and DTSVM classifiers respectively, and the values of R/L. Training types = 1A1 and 1AO.

gSVM classifiers (i.e., in (1) and (2)) with the common factor $\log^2 n$ removed from them. To ensure a meaningful comparison between R and S, both classifiers have to take the same (C,γ) values, which we specify as the optimal values for gSVM. As a result, we had to train new DTSVM classifiers for some data sets, using the same decomposition schemes (i.e., the same binary trees and same ceiling sizes) as the old classifiers, but different (C,γ) values.

Table 14 shows the values of S, R and R/L, derived from five data sets. The "Letter" and "News20" data sets are not included in the table because the DTSVM classifier using the designated values of (C,γ) would be the same as the gSVM on those data sets. It is clear that the values of R are as small as (less than 150), or of the same order of magnitude as, those of the corresponding S. In fact, S can be viewed as the slack-to-margin ratio and R as the sum of such ratios. The results show that each lSVM generates smaller slack-to-margin ratios than the corresponding gSVM, while the sum of lSVM ratios is comparable to the corresponding gSVM ratio. This explains why the test accuracy rates of DTSVM classifiers are comparable to those of gSVM classifiers.

5. Conclusion

We have proposed a method that uses a binary tree to decompose a given data space and trains an lSVM on each of the decomposed regions. The resultant DTSVM classifier can be constructed in a much shorter time than the gSVM classifier, and still achieve comparable accuracy rates to the latter. We also provide a generalization error bound for the DTSVM classifier. Using some data sets to compute the theoretical bounds for gSVM and DTSVM classifiers, we find that DTSVM classifiers generate comparable error bounds to those generated by gSVM classifiers. This finding explains why DTSVM classifiers can achieve more or less the same accuracy rates as gSVM classifiers.

Appendix A. Proof of Lemma 4

Let $\mathcal{G} \subseteq \mathcal{P}_L(\mathbf{R}^d)$, $\gamma = (\gamma_1,\ldots,\gamma_L) \in (\mathbf{R}^+)^L$, and $\mathcal{F}_i \subseteq \mathcal{R}(\mathbf{H})$ for $i = 1,\ldots,L$. The samples in X_n are drawn independently and at random according to the distribution $\mathcal{D}$.

Our goal is to find an upper bound for the probability of the following event.

- A_1: there exist $\pi \in \mathcal{G}$ and $\mathbf{f} = (f_1, \ldots, f_L) \in \mathcal{F}_1 \times \cdots \times \mathcal{F}_L$ such that $mg(\mathbf{f}^\pi, X_n) \geq \gamma$ and $err(\mathbf{f}^\pi, \mathcal{D}) > \varepsilon$, where $err(\mathbf{f}^\pi, \mathcal{D})$ is the probability that $sign(\mathbf{f}^\pi(\mathbf{x})) \neq y$ with $(\mathbf{x}, y)$ being sampled according to $\mathcal{D}$.

Note that $err(\mathbf{f}^\pi, \mathcal{D})$ is the generalization error of $sign(\mathbf{f}^\pi(\mathbf{x}))$. We relate event A_1 to another event A_2 in which an additional set of n independent samples, $\hat{X}_n$, are drawn at random according to $\mathcal{D}$ and the empirical error of $sign(\mathbf{f}^\pi(\mathbf{x}))$ over $\hat{X}_n$ is considered.

- A_2: there exist $\pi \in \mathcal{G}$ and $\mathbf{f} = (f_1, \ldots, f_L) \in \mathcal{F}_1 \times \cdots \times \mathcal{F}_L$ such that $mg(\mathbf{f}^\pi, X_n) \geq \gamma$ and $err(\mathbf{f}^\pi, \hat{X}_n) > \varepsilon/2$, where $err(\mathbf{f}^\pi, \hat{X}_n) = |\{(x, y) \in \hat{X}_n : sign(\mathbf{f}^\pi(\mathbf{x})) \neq y\}|/n$.

Following a standard argument (Vapnik, 1995), one can relate the probability of A_1 with that of A_2. More precisely, we have

$$\Pr_{X_n, \hat{X}_n}[A_2] \geq \Pr_{X_n, \hat{X}_n}[A_2 \wedge A_1] = \Pr_{X_n}[A_1] \cdot \Pr_{X_n, \hat{X}_n}[A_2 | A_1],$$

and by Chebyshev's inequality, one can show that

$$\Pr_{X_n, \hat{X}_n}[A_2 | A_1] = 1 - \Pr_{X_n, \hat{X}_n}[\neg A_2 | A_1] \geq 1 - 1/(n\varepsilon^2) \geq 1/2,$$

for a large enough n. Consequently, we have

$$\Pr_{X_n}[A_1] \leq \left(1 / \Pr_{X_n, \hat{X}_n}[A_2 | A_1]\right) \cdot \Pr_{X_n, \hat{X}_n}[A_2] \leq 2 \cdot \Pr_{X_n, \hat{X}_n}[A_2].$$

To find a bound for $\Pr_{X_n, \hat{X}_n}[A_2]$, let us consider the following event, A_3, where a set of $2n$ samples, X_{2n}, are drawn independently and at random according to $\mathcal{D}$, and X_{2n} is further divided randomly into two disjoint parts of equal size: W_1 and W_2.

- A_3: there exist $\pi \in \mathcal{G}$ and $\mathbf{f} = (f_1, \ldots, f_L) \in \mathcal{F}_1 \times \cdots \times \mathcal{F}_L$ such that $mg(\mathbf{f}^\pi, W_1) \geq \gamma$ and $err(\mathbf{f}^\pi, W_2) > \varepsilon/2$.

We observe that the distribution of $(X_n, \hat{X}_n)$ is identical to that of (W_1, W_2) over random X_{2n}; consequently we have

$$\Pr_{X_{2n}, W_1, W_2}[A_3] = \Pr_{X_n, \hat{X}_n}[A_2].$$

The next step is to find a bound for $\Pr_{X_{2n}, W_1, W_2}[A_3]$.

Let $\mathcal{G}(X_{2n})$ be the family of functions of $\mathcal{G}$ restricted to the domain $\{\mathbf{x} : (\mathbf{x}, y) \in X_{2n}\}$; and for $\pi \in \mathcal{G}(X_{2n})$, let

$$B^\pi_{\gamma/2}(X_{2n}) = \mathcal{C}\left(\mathcal{F}_1, \Phi(X^{(1)}_{2n}), \gamma_1/2\right) \times \cdots \times \mathcal{C}\left(\mathcal{F}_L, \Phi(X^{(L)}_{2n}), \gamma_L/2\right),$$

where $\Phi(X^{(i)}_{2n}) = \{\Phi(\mathbf{x}) : (\mathbf{x}, y) \in X_{2n}, \pi(\mathbf{x}) = i\}$. For $\mathbf{g} = (g_1, \ldots, g_L) \in B^\pi_{\gamma/2}(X_{2n})$, let $g^\pi(\mathbf{x}) = g_i(\mathbf{x})$ for $\mathbf{x} \in \Phi(X^{(i)}_{2n})$ for $1 \leq i \leq L$. Then, for $\pi \in \mathcal{G}(X_{2n})$ and $\mathbf{f} = (f_1, \ldots, f_n) \in \mathcal{F}_1 \times \cdots \times \mathcal{F}_L$, there exists $\mathbf{g} = (g_1, \ldots, g_L) \in B^\pi_{\gamma/2}(X_{2n})$ such that for any $(\mathbf{x}, y) \in X_n$, if $\pi(\mathbf{x}) = i$, then

$$|f_i(\Phi(\mathbf{x})) - g_i(\Phi(\mathbf{x}))| \leq \gamma_i/2.$$

For such $\mathbf{f}^\pi$ and $\mathbf{g}^\pi$, $mg(\mathbf{f}^\pi, W_1) \geq \gamma$ implies that $mg(\mathbf{g}^\pi, W_1) \geq \gamma/2$; and $err(\mathbf{f}^\pi, W_2) \geq \varepsilon/2$ implies that $err_{\gamma/2}(\mathbf{g}^\pi, W_2) \geq \varepsilon/2$, where $err_{\gamma/2}(\mathbf{g}^\pi, W_2)$ is the proportion of $(\mathbf{x}, y)$ in W_2 for which $g_i(\mathbf{x}) < \gamma_i/2$ if $\pi(\mathbf{x}) = i$. Therefore, the probability of the event A_3 cannot exceed that of the following event A_4.

- A_4: there exist $\pi \in \mathcal{G}(X_{2n})$ and $\mathbf{g} \in B^\pi_{\gamma/2}(X_{2n})$ such that $mg(\mathbf{g}^\pi, W_1) \geq \gamma/2$ and $err_{\gamma/2}(\mathbf{g}^\pi, W_2) \geq \varepsilon/2$.

To bound the probability of A_4, let us first consider any fixed X_{2n}, π, and $\mathbf{g}$, and consider an event, denoted as $A_4(X_{2n}, \pi, \mathbf{g})$, over the random division of X_{2n} into W_1 and W_2, such that $mg(\mathbf{g}^\pi, W_1) \geq \gamma/2$ and $err_{\gamma/2}(\mathbf{g}^\pi, W_2) \geq \varepsilon/2$. Note that for the event $A_4(X_{2n}, \pi, \mathbf{g})$ to occur, there are at most $2n - (\varepsilon/2)n$ elements of X_{2n} which can be separated by $\mathbf{g}^\pi$ with margin $\gamma/2$, and these elements must contain all the n elements of W_1. This implies that

$$\Pr_{W_1, W_2}[A_4(X_{2n}, \pi, \mathbf{g})] \leq \frac{\binom{2n-(\varepsilon/2)n}{n}}{\binom{2n}{n}} \leq \left(\frac{n}{2n}\right)^{(\varepsilon/2)n} = 2^{-\varepsilon n/2}.$$

Next, let us fix any X_{2n} and consider an event, denoted as $A_4(X_{2n})$, over the random division of X_{2n} into W_1 and W_2, such that the event $A_4(X_{2n}, \pi, \mathbf{g})$ occurs for some $\pi \in \mathcal{G}(X_{2n})$ and $\mathbf{g} \in B^\pi_{\gamma/2}(X_{2n})$. By a simple union bound, we have

$$\Pr_{W_1, W_2}[A_4(X_{2n})] \leq \sum_{\pi \in \mathcal{G}(X_{2n})} \sum_{g \in B^\pi_{\gamma/2}(X_{2n})} 2^{-\varepsilon n/2} = |\mathcal{G}(X_{2n})| \cdot |B^\pi_{\gamma/2}(X_{2n})| \cdot 2^{-\varepsilon n/2}.$$

Since $|X_{2n}| = 2n$, we have $|\mathcal{G}(X_{2n})| \leq V(\mathcal{G}, 2n)$. Moreover,

$$\begin{aligned}
\left|B^\pi_{\gamma/2}(X_{2n})\right| &= \prod_{1 \leq i \leq L} \left|C\left(\mathcal{F}_i, X^{(i)}_{2n}, \gamma_i/2\right)\right| \\
&\leq \prod_{1 \leq i \leq L} N\left(\mathcal{F}_i, E, |X^{(i)}_{2n}|, \gamma_i/2\right) \\
&\leq \prod_{1 \leq i \leq L} N(\mathcal{F}_i, E, 2n, \gamma_i/2),
\end{aligned}$$

where $E = \{\Phi(\mathbf{x}) : (\mathbf{x}, y) \in supp(\mathcal{D})\}$. Therefore, we have

$$\Pr_{W_1, W_2}[A_4(X_{2n})] \leq V(\mathcal{G}, 2n) \cdot \left(\prod_{1 \leq i \leq L} N(\mathcal{F}_i, E, 2n, \gamma_i/2)\right) \cdot 2^{-\varepsilon n/2}.$$

Note that by randomizing the selection of X_{2n}, the expected value of $\Pr_{W_1, W_2}[A_4(X_{2n})]$ is just the probability of A_4, over the random selection of X_{2n} and its random division into W_1 and W_2, from a simple probability fact.[1] As a result, we have

$$\begin{aligned}
\Pr_{X_{2n}, W_1, W_2}[A_4] &= \mathop{\mathrm{E}}_{X_{2n}}\left[\Pr_{W_1, W_2}[A_4(X_{2n})]\right] \\
&\leq V(\mathcal{G}, 2n) \cdot \left(\prod_{1 \leq i \leq L} N(\mathcal{F}_i, E, 2n, \gamma_i/2)\right) \cdot 2^{-\varepsilon n/2}.
\end{aligned}$$

1. Suppose A is an event over a joint distribution $(\mathbf{X}, \mathbf{W})$. Let $A(X)$ denote the event A conditioned on $\mathbf{X} = X$. Then, $\Pr_{(X,W) \in (\mathbf{X}, \mathbf{W})}[A] = \sum_Z \Pr_{X \in \mathbf{X}}[X = Z] \cdot \Pr_{W \in \mathbf{W}}[A(Z)] = \mathrm{E}_{X \in \mathbf{X}}[\Pr_{W \in \mathbf{W}}[A(X)]]$.

Finally, by combining all the bounds derived so far, we obtain

$$\Pr_{X_n}[A_1] \leq 2 \cdot \Pr_{X_{2n},W_1,W_2}[A_4] \leq 2 \cdot V(\mathcal{G},2n) \cdot \left(\prod_{1 \leq i \leq L} N(\mathcal{F}_i, E, 2n, \gamma_i/2) \right) \cdot 2^{-\varepsilon n/2},$$

which is at most d if

$$\varepsilon = \frac{2}{n} \left(\log V(\mathcal{G},2n) + \sum_{1 \leq i \leq L} \log N(\mathcal{F}_i, E, 2n, \gamma_i/2) + \log(2/\delta) \right).$$

This proves the lemma.

Appendix B. Proof of Lemma 9

We begin by describing the space $\hat{\mathbf{H}}$ and the mapping $\tau_\rho : \mathbf{H} \to \hat{\mathbf{H}}$. Consider the inner product space

$$I(\mathbf{H}) = \{ f \in \mathcal{R}(\mathbf{H}) : f(\mathbf{x}) \neq 0 \text{ for a finite number of } \mathbf{x} \in \mathbf{H} \},$$

where the inner product of f and g in $I(\mathbf{H})$ is defined as $\langle f, g \rangle = \sum_z f(z)g(z)$. Let

$$\hat{\mathbf{H}} = \mathbf{H} \times I(\mathbf{H}).$$

Define the function $\tau_\rho : \mathbf{H} \to \hat{\mathbf{H}}$ by

$$\tau_\rho(z) = (z, \rho \cdot \delta_z),$$

where δ_z is the function defined by

$$\delta_z(z') = \begin{cases} 1 & \text{if } z = z', \\ 0 & \text{otherwise.} \end{cases}$$

Then, the first item of the lemma holds, since for $(\mathbf{x}, y) \in supp(\mathcal{D})$, $\|\Phi(\mathbf{x})\| \leq \rho$; thus,

$$\|\tau_\rho(\Phi(x))\|^2 = \|\Phi(x)\|^2 + \|\rho \delta_{\Phi(x)}\|^2 \leq 2\rho^2.$$

Next, we prove the existence of $\hat{\mathbf{f}} = (\hat{f}_1, \ldots, \hat{f}_L)$. Since $I(\mathbf{H})$ is an inner product space, each of its elements defines a linear function on $I(\mathbf{H})$. Hence, for $(f,g) \in \mathcal{L}(\mathbf{H}) \times I(\mathbf{H})$, the function $(f,g) : \hat{\mathbf{H}} \to \mathbf{R}$, defined by

$$(f,g)(z,h) = f(z) + \langle g, h \rangle,$$

for $(z,h) \in \hat{\mathbf{H}}$, is a linear function. Now, for $1 \leq i \leq L$, we can define $\hat{f}_i : \hat{\mathbf{H}} \to \mathbf{R}$ by

$$\hat{f}_i = (f_i, g_i/\rho),$$

where $g_i \in I(\mathbf{H})$ is defined by

$$g_i = \sum_{j=1}^{n_i} \xi_{i,j} \cdot y_{i,j} \cdot \delta_{\Phi(\mathbf{x}_{i,j})}.$$

Since $f_i \in \mathcal{L}(\mathbf{H}, \beta_i)$, there exists $w_i \in \mathbf{H}$ with $\|w_i\| \leq \beta_i$ such that $f_i(z) = \langle w_i, z \rangle$. It follows that for $\hat{z} \in \hat{\mathbf{H}}$,

$$\hat{f}_i(\hat{z}) = \langle \hat{w}_i, \hat{z} \rangle,$$

where $\hat{w}_i = (w_i, g_i/\rho)$, and

$$\|\hat{w}_i\|^2 = \|w_i\|^2 + \sum_{j=1}^{n_i} |\xi_{i,j}|^2/\rho^2 \leq \beta_i^2 + \|\xi_i\|^2/\rho^2.$$

Therefore, $\hat{f}_i \in \mathcal{L}(\hat{\mathbf{H}}, \hat{\beta}_i)$, where $\hat{\beta}_i = \sqrt{\beta_i^2 + \|\xi_i\|^2/\rho^2}$, and the second item of the lemma holds.

Furthermore, for any $(\mathbf{x}_{i,j}, y_{i,j}) \in X_n^{(i)}$, we have

$$\begin{aligned} y_{i,j} \cdot \hat{f}_i(\tau_\rho(\Phi(\mathbf{x}_{i,j}))) &= y_{i,j} \cdot f_i(\Phi(\mathbf{x}_{i,j})) + y_{i,j} \cdot (\xi_{i,j} \cdot y_{i,j}) \\ &= y_{i,j} \cdot f_i(\Phi(\mathbf{x}_{i,j})) + \xi_{i,j} \\ &\geq \gamma_i, \end{aligned}$$

by the definition of $\xi_{i,j}$. Thus, the third item of the lemma also holds.

Finally, for $1 \leq i \leq L$ and for any $(\mathbf{x}, y) \notin X_n$, we have

$$\hat{f}_i(\tau_\rho(\Phi(\mathbf{x}))) = f_i(\Phi(\mathbf{x})) + \sum_{j=1}^{n_i} \xi_{i,j} \cdot y_{i,j} \cdot \delta_{\Phi(\mathbf{x}_{i,j})}(\Phi(\mathbf{x})) = f_i(\Phi(\mathbf{x})).$$

Thus, the fourth item of the lemma holds as well. This concludes the proof of Lemma 9.

References

N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of ACM*, 44(4):615–631, 1997.

P. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

K. P. Bennett and J. A. Blue. A support vector machine approach to decision trees. In *Proceedings IEEE International Joint Conference on Neural Networks*, 1998.

K. P. Bennett, N. Cristianini, J. Shawe-Taylor, and D. Wu. Enlarging the margins in perceptron decision trees. *Machine Learning*, 41:295–313, 2000.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *Proceedings of the 24th International Machine Learning Conference*, pages 89–96, 2007.

A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.

L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Müller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: A case study in handwriting digit recognition. In *Proc. Int. Conf. Pattern Recognition*, pages 77–87, 1994.

L. Breiman. Bagging predictors. *Machine Learning*, 262:123–140, 1996.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall, 1984.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Journal of Data Mining and Knowledge Discovery*, 2(2):1–47, 1998.

R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(5):1105–1114, 2002.

C. Cortes and V. Vapnik. Support vector machines. *Machine Learning*, 20:1–25, 1995.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

G. R. Dattatreya and L. N. Kanal. Decision trees in pattern recognition. In L. N. Kanal and A. Rossenfeld, editors, *Progress in Pattern Recognition*, volume 2. Elsevier, 1985.

T. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157, 2000.

R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th International Machine Learning Conference*, pages 320–327, 2008.

T. Glasmachers and C. Igle. Maximum-gain working set selection for SVMs. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik. Parallel support vector machines: the cascade SVM. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Neural Information Processing Systems*. MIT Press, 2004.

T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Kernel Methods – Support Vector Learning*. MIT Press, 1998.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, 2006.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8), 2004.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing: Algorithms, Architectures and Applications*. Springer-Verlag, 1990.

N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: the informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Neural Information Processing Systems*. MIT Press, 2003.

A. K. Menon. Large-scale support vector machines: algorithms and theory. Research Exam, University of California, San Diego, 2009.

S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Machine Learning Research*, 2:1–32, 1994.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

E. Osuna, R. Freud, and F. Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing*, volume VII, pages 276–285, 1997.

N. Panda, E. Y. Chang, and G. Wu. Concept boundary detection for speeding up SVMs. In *Proceedings of International Conference on Machine learning*, pages 681–688, 2006.

D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *ACM SIGKDD*, pages 295–299, 2000.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. J. Smola, editors, *Kernel Methods: Support Vector Learning*. MIT Press, 1998.

J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2000.

J. R. Quinlan. Induction of decision tree. *Machine Learning*, 1:81–106, 1986.

S. Ramaswamy. Multiclass text classification a decision tree based SVM approach. CS294 Practical Machine Learning Project, 2006.

A. Rida, A. Labbi, and C. Pellegrini. Local experts combination through density decomposition. In *Proceedings of International Workshop on AI and Statistics*, 1999.

H. Sahbi and D. Geman. A hierarchy of support vector machines for pattern detection. *Journal of Machine Learning Research*, 7:2087–2123, 2006.

R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Journal of Machine Learning Research*, 39(2/3):135–168, 2000.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of International Conference on Machine learning*, pages 807–814, 2007.

J. Shawe-Taylor and N. Cristianini. Margin distribution bounds on generalization. In *Proceedings of the European Conference on Computational Learning Theory*, pages 263–273, 1999.

J. Shawe-Taylor and N. Cristianini. On the generalization of soft margin algorithms. *IEEE Transactions on Information Theory*, 48(10):2721–2735, 2002.

A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the International Conference on Machine Learning*, pages 911–918, 2000.

A. J. Smola, S. V. N. Vishwanathan, and Q. V. Le. Bundle methods for machine learning. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. MIT Press, 2007.

J.-Q. Sun, G.-G. Wang, Q. Hu, and S.-Y. Li. A novel SVM detection tree and its application to face detection. In *Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pages 385–389, 2007.

R. Tibshirani and T. Hastie. Margin trees for high-dimensional classification. *Journal of Machine Learning Research*, 8:637–652, 2007.

V. Tresp. A bayesian committee machines. *Neural Computation*, 12:2719–2741, 2000.

V. Tresp. Scaling kernel-based systems to large data sets. *Data Mining and Knowledge Discovery*, 5:197–211, 2001.

I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

T.-S. Tseng, C.-Y. Guo, Wen-Lian Hsu, and F. Chang. Protein-protein interface prediction based on a novel SVM speedup. Technical Report, Number TR-IIS-10-001, Institute of Information Science, Academia Sinica, 2010.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

D. Wu, K. P. Bennett, N. Cristianini, and J. Shawe-Taylor. Large margin decision trees for induction and transduction. In *Proceedings of International Conference on Machine Learning*, pages 474–483, 1999.

H. Yu, J. Han J. Yang, and X. Li. Making SVMs scalable to large data sets using hierarchical cluster indexing. *Data Mining and Knowledge Discovery*, 11:295–321, 2003.

Journal of Machine Learning Research 11 (2010) 2973-3009 Submitted 7/09; Revised 5/10; Published 11/10

Semi-Supervised Novelty Detection*

Gilles Blanchard BLANCHARD@WIAS-BERLIN.DE
Universität Potsdam, Intitut für Mathematik
Am Neuen Palais 10
14469 Potsdam, Germany

Gyemin Lee GYEMIN@EECS.UMICH.EDU
Clayton Scott CSCOTT@EECS.UMICH.EDU
Department of Electrical Engineering and Computer Science
University of Michigan
1301 Beal Avenue
Ann Arbor, MI 48109-2122, USA

Editor: Ingo Steinwart

Abstract

A common setting for novelty detection assumes that labeled examples from the nominal class are available, but that labeled examples of novelties are unavailable. The standard (inductive) approach is to declare novelties where the nominal density is low, which reduces the problem to density level set estimation. In this paper, we consider the setting where an unlabeled and possibly contaminated sample is also available at learning time. We argue that novelty detection in this semi-supervised setting is naturally solved by a general reduction to a binary classification problem. In particular, a detector with a desired false positive rate can be achieved through a reduction to Neyman-Pearson classification. Unlike the inductive approach, semi-supervised novelty detection (SSND) yields detectors that are optimal (e.g., statistically consistent) regardless of the distribution on novelties. Therefore, in novelty detection, unlabeled data have a substantial impact on the theoretical properties of the decision rule. We validate the practical utility of SSND with an extensive experimental study.

We also show that SSND provides distribution-free, learning-theoretic solutions to two well known problems in hypothesis testing. First, our results provide a general solution to the general two-sample problem, that is, the problem of determining whether two random samples arise from the same distribution. Second, a specialization of SSND coincides with the standard p-value approach to multiple testing under the so-called random effects model. Unlike standard rejection regions based on thresholded p-values, the general SSND framework allows for adaptation to arbitrary alternative distributions in multiple dimensions.

Keywords: semi-supervised learning, novelty detection, Neyman-Pearson classification, learning reduction, two-sample problem, multiple testing

1. Introduction

Several recent works in the machine learning literature have addressed the issue of novelty detection. The basic task is to build a decision rule that distinguishes *nominal* from *novel* patterns. The learner is given a random sample $x_1, \ldots, x_m \in \mathcal{X}$ of nominal patterns, obtained, for example, from a

*. A preliminary version of this work appeared at AISTATS (Scott and Blanchard, 2009).

controlled experiment or an expert. Labeled examples of novelties, however, are not available. The standard approach has been to estimate a level set of the nominal density (Schölkopf et al., 2001; Steinwart et al., 2005; Scott and Nowak, 2006; Vert and Vert, 2006; El-Yaniv and Nisenson, 2007; Hero, 2007), and to declare test points outside the estimated level set to be novelties. We refer to this approach as *inductive* novelty detection.

In this paper we incorporate unlabeled data into novelty detection, and argue that this framework offers substantial advantages over the inductive approach. In particular, we assume that in addition to the nominal data, we also have access to an *unlabeled* sample $x_{m+1},\ldots,x_{m+n}$ consisting potentially of both nominal and novel data. We assume that each x_i, $i = m+1,\ldots,m+n$ is paired with an unobserved label $y_i \in \{0,1\}$ indicating its status as nominal ($y_i = 0$) or novel ($y_i = 1$), and that $(x_{m+1},y_{m+1}),\ldots,(x_{m+n},y_{m+n})$ are realizations of the random pair (X,Y) with joint distribution P_{XY}. The marginal distribution of an unlabeled pattern X is the contamination model

$$X \sim P_X = (1-\pi)P_0 + \pi P_1,$$

where P_y, $y = 0,1$, is the conditional distribution of $X|Y = y$, and $\pi = P_{XY}(Y = 1)$ is the a priori probability of a novelty. Similarly, we assume $x_1,\ldots,x_m$ are realizations of P_0. We assume no knowledge of P_X, P_0, P_1, or π, although in Section 6 (where we want to estimate the proportion π) we do impose a natural condition on P_1 that ensures identifiability of π.

We take as our objective to build a decision rule with a small false negative rate subject to a fixed constraint α on the false positive rate. Our emphasis here is on *semi-supervised* novelty detection (SSND), where the goal is to construct a general detector that could classify an arbitrary test point. This general detector can of course be applied in the *transductive* setting, where the goal is to predict the labels $y_{m+1},\ldots,y_{m+n}$ associated with the unlabeled data. Our results extend in a natural way to this setting.

Our basic contribution is to develop a general solution to SSND by a surrogate problem related to Neyman-Pearson (NP) classification, which is the problem of binary classification subject to a user-specified constraint α on the false positive rate. In particular, we argue that SSND can be addressed by applying a NP classification algorithm, treating the nominal and unlabeled samples as the two classes. Even though a sample from P_1 is not available, we argue that our approach can effectively adapt to any novelty distribution P_1, in contrast to the inductive approach which is only optimal in certain extremely unlikely scenarios. That is, by solving the surrogate problem, we obtain a classifier f such that, up to a tolerance that shrinks as sample sizes increase, $P_1(f(X) = 0)$ is minimal, while $P_0(f(X) = 1) \leq \alpha$.

Our learning reduction allows us to import existing statistical performance guarantees for Neyman-Pearson classification (Cannon et al., 2002; Scott and Nowak, 2005) and thereby deduce generalization error bounds, consistency, and rates of convergence for novelty detection. In addition to these theoretical properties, the reduction to NP classification has practical advantages, in that it allows essentially any algorithm for NP classification to be applied to SSND.

SSND is particularly suited to situations where the novelties occupy regions where the nominal density is high. If a single novelty lies in a region of high nominal density, it will appear nominal. However, if many such novelties are present, the unlabeled data will be more concentrated than one would expect from just the nominal component, and the presence of novelties can be detected. SSND may also be thought of as semi-supervised classification in the setting where labels from one class are difficult to obtain (see discussion of LPUE below). We emphasize that we do not assume

that novelties are rare, that is, that π is very small, as in anomaly detection. However, SSND is applicable to anomaly detection provided n is sufficiently large.

We also discuss estimation of π and the special case of $\pi = 0$, which is not treated in our initial analysis. We present a hybrid approach that automatically reverts to the inductive approach when $\pi = 0$, while preserving the benefits of the NP reduction when $\pi > 0$. In addition, we describe a distribution-free one-sided confidence interval for π, consistent estimation of π, and testing for $\pi = 0$, which amounts to a general version of the two-sample problem in statistics. We also discuss connections to multiple testing, where we show that SSND generalizes a standard approach to multiple testing, based on thresholding p-values, under the common "random effects" model. Whereas the p-value approach is optimal only under strong assumptions on the alternative distribution, SSND can optimally adapt to arbitrary alternatives.

The paper is structured as follows. After reviewing related work in the next section, we present the general learning reduction to NP classification in Section 3, and apply this reduction in Section 4 to deduce statistical performance guarantees for SSND. Section 5 presents our hybrid approach, while Section 6 applies learning-theoretic principles to inference on π. Connections to multiple testing are developed in Section 7. Experiments are presented in Section 8, while conclusions are discussed in the final section. Shorter proofs are presented in the main text, and longer proofs appear in the first appendix.

2. Related Work

Inductive novelty detection: Described in the introduction, this problem is also known as one-class classification (Schölkopf et al., 2001) or learning from only positive (or only negative) examples. The standard approach has been to assume that novelties are outliers with respect to the nominal distribution, and to build a novelty detector by estimating a level set of the nominal density (Scott and Nowak, 2006; Vert and Vert, 2006; El-Yaniv and Nisenson, 2007; Hero, 2007). As we discuss below, density level set estimation is equivalent to assuming that novelties are uniformly distributed on the support of P_0. Therefore these methods can perform arbitrarily poorly (when P_1 is far from uniform, and still has significant overlap with P_0). In Steinwart et al. (2005), inductive novelty detection is reduced to classification of P_0 against P_1 , wherein P_1 can be arbitrary. However an i.i.d. sample from P_1 is assumed to be available in addition to the nominal data. In contrast, the semi-supervised approach optimally adapts to P_1, where only an unlabeled contaminated sample is available besides the nominal data. In addition, we address estimation and testing of the proportion of novelties.

Classification with unlabeled data: In transductive and semi-supervised classification, labeled training data $\{(x_i, y_i)\}_{i=1}^m$ from *both* classes are given. The setting proposed here is a special case where training data from only one class are available. In two-class problems, unlabeled data typically have at best a slight effect on constants, finite sample bounds, and rates (Rigollet, 2007; Lafferty and Wasserman, 2008; Ben-David et al., 2008; Singh et al., 2009), and are not needed for consistency. In contrast, we argue that for novelty detection, unlabeled data are essential for these desirable theoretical properties to hold.

Learning from positive and unlabeled examples: Classification of an unlabeled sample given data from one class has been addressed previously, but with certain key differences from our work. This body of work is often termed learning from "positive" and unlabeled examples (LPUE), al-

though in our context we tend to think of nominal examples as negative. Terminology aside, a number of algorithms have been developed, which we now relate to the present work.

One class of algorithms proceeds roughly as follows: First, identify unlabeled points for which it seems highly likely that $y_i = 1$. Second, learn a classifier from the known positive examples and the supposed negative examples. Use it on the unlabeled data to update the group of candidates for the negative class and repeat until a stable labeling is reached. Several such algorithms are reviewed in Zhang and Lee (2005) and Zhang and Zuo (2008), but they tend to be heuristic in nature and sensitive to the initial choice of negative examples.

A theoretical analysis of LPUE is provided by Denis (1998); Denis et al. (2005) from the viewpoint of probably approximately correct (PAC) learnable classes. In PAC learnability, the objective is to find specific classes of classifiers such that the optimal classifier in that class can be approximated arbitrarily well, and where the number of samples required is polynomial in the inverse of the error tolerance. While some ideas are common with the present work (such as classifying the nominal sample against the contaminated sample as a proxy for the ultimate goal), our point of view is relatively different and based on statistical learning theory. In particular, our input space can be non-discrete and we assume the distributions P_0 and P_1 can overlap, which leads us to use the NP classification setting and study universal consistency properties.

Several other approaches have been developed which, either explicitly or implicitly, rely on a reduction to a classification problem. Steinberg and Cardell (1992) and Ward et al. (2009) propose frameworks based on logistic regression, but both assume that π is known. Elkan and Noto (2008) assume a particular sampling scheme where m and n are related in such a way that π can be readily estimated. Unfortunately, this sampling assumption is not valid in many applications of interest. All three of these works derive their algorithms by a consideration of posterior probabilities, and consequently they require that π is known or can be estimated. In contrast, our approach adopts the (non-Bayesian) Neyman-Pearson criterion and in no way depends on the ability to know or estimate π.

The idea of reducing LPUE to a binary classification problem has also been treated by Zhang and Lee (2005), Liu et al. (2002), Lee and Liu (2003) and Liu et al. (2003). Most notably, Liu et al. (2002) provide sample complexity bounds for VC classes for the learning rule that minimizes the number of false negatives while controlling the proportion of false positives at a certain level. Our approach extends theirs in several respects. First, Liu et al. (2002) does not consider approximation error or consistency, nor do the bounds established there imply consistency. In contrast, we present a general reduction that is not specific to any particular learning algorithm, and can be used to deduce consistency or rates of convergence. Our work also makes several contributions not addressed previously in the LPUE literature, including our results relating to the case $\pi = 0$, to the estimation of π, and to multiple testing.

We also note recent work by Smola et al. (2009) described as *relative novelty detection*. This work is presented as an extension of standard one-class classification to a setting where a reference measure (indicating regions where novelties are more likely) is known through a sample. In practice, the authors take this sample to be a contaminated sample consisting of both nominal and novel measurements, so the setting is the same as ours. The emphasis in this work is primarily on a new kernel method, whereas our work features a general learning reduction and learning theoretic analysis.

Multiple testing: The multiple testing problem is also concerned with the simultaneous detection of many potentially abnormal measurements (viewed as rejected null hypotheses). In Section 7, we

discuss in detail the relation of our contamination model to the *random effects model*, a standard model in multiple testing. We show how SSND is, in several respects, a generalization of that model, and includes several different extensions proposed in the recent multiple testing literature. The SSND model, and the results presented in this paper, are thus relevant to multiple testing as well, and suggest an interesting point of view to this domain. In particular, through a reduction to classification, we introduce broad connections to statistical learning theory.

3. The Fundamental Reduction

To begin, we first consider the population version of the problem, where the distributions are known completely. Recall that $P_X = (1-\pi)P_0 + \pi P_1$ is the distribution of unlabeled test points. Adopting a hypothesis testing perspective, we argue that the optimal tests for $H_0 : X \sim P_0$ vs. $H_1 : X \sim P_1$ are identical to the optimal tests for $H_0 : X \sim P_0$ vs. $H_X : X \sim P_X$. The former are the tests we would like to have, and the latter are tests we can estimate by treating the nominal and unlabeled samples as labeled training data for a binary classification problem.

To offer some intuition, we first assume that P_y has density h_y, $y = 0, 1$. According to the Neyman-Pearson Lemma (Lehmann, 1986), the optimal test with size (false positive rate) α for $H_0 : X \sim P_0$ vs. $H_1 : X \sim P_1$ is given by thresholding the likelihood ratio $h_1(x)/h_0(x)$ at an appropriate value. Similarly, letting $h_X = (1-\pi)h_0 + \pi h_1$ denote the density of P_X, the optimal tests for $H_0 : X \sim P_0$ vs. $H_X : X \sim P_X$ are given by thresholding $h_X(x)/h_0(x)$. Now notice

$$\frac{h_X(x)}{h_0(x)} = (1-\pi) + \pi \frac{h_1(x)}{h_0(x)}.$$

Thus, the likelihood ratios are related by a simple monotone transformation, provided $\pi > 0$. Furthermore, the two problems have the same null hypothesis. Therefore, by the theory of uniformly most powerful tests (Lehmann, 1986), the optimal test of size α for one problem is also optimal, *with the same size* α, for the other problem. In other words, we can discriminate P_0 from P_1 by discriminating between the nominal and unlabeled distributions. Note the above argument does not require knowledge of π other than $\pi > 0$.

The hypothesis testing perspective also sheds light on the inductive approach. In particular, estimating the nominal level set $\{x : h_0(x) \geq \lambda\}$ is equivalent to thresholding $1/h_0(x)$ at $1/\lambda$. Thus, the density level set is an optimal decision rule provided h_1 is constant on the support of h_0. This assumption that P_1 is uniform on the support of P_0 is therefore implicitly adopted by a majority of works on novelty detection.

We now drop the requirement that P_0 and P_1 have densities. Let $f : \mathbb{R}^d \to \{0, 1\}$ denote a classifier. For $y = 0, 1$, let

$$R_y(f) := P_y(f(X) \neq y)$$

denote the false positive rate (FPR) and false negative rate (FNR) of f, respectively. For greater generality, suppose we restrict our attention to some fixed set of classifiers $\mathcal{F}$ (possibly the set of all classifiers). The optimal FNR for a classifier of the class $\mathcal{F}$ with FPR $\leq \alpha, 0 \leq \alpha \leq 1$, is

$$\begin{aligned} R^*_{1,\alpha}(\mathcal{F}) \quad := \quad & \inf_{f \in \mathcal{F}} R_1(f) \\ & \text{s.t. } R_0(f) \leq \alpha. \end{aligned} \tag{1}$$

Similarly, introduce

$$\begin{aligned} R_X(f) &:= P_X(f(X)=0) \\ &= \pi R_1(f) + (1-\pi)(1-R_0(f)) \end{aligned}$$

and let

$$\begin{aligned} R^*_{X,\alpha}(\mathcal{F}) &:= \inf_{f\in\mathcal{F}} R_X(f) \\ &\quad \text{s.t. } R_0(f) \le \alpha. \end{aligned} \tag{2}$$

In this paper we will always assume the following property (involving $\mathcal{F}, P_0$ and P_1) holds:

(A) For any $\alpha \in (0,1)$, there exists $f^* \in \mathcal{F}$ such that $R_0(f^*) = \alpha$ and $R_1(f^*) = R^*_{1,\alpha}(\mathcal{F})$.

Remark. *This assumption is in particular satisfied if the class $\mathcal{F}$ is such that for any $f \in \mathcal{F}$ with $R_0(f) < \alpha$, we can find another classifier $f' \in \mathcal{F}$ with $R_0(f') = \alpha$ and $f' \ge f$ (so that $R_1(f') \le R_1(f)$). When P_0 is absolutely continuous with respect to Lebesque measure, this property can be easily verified for many common classifier sets, for example linear classifiers, decision trees or radial basis function classifiers.*

Even without any assumptions on the distribution, it is possible to ensure that **(A)** *is satisfied provided one extends the class $\mathcal{F}$ to a larger class containing randomized classifiers obtained by convex combination of classifiers of the original class. This construction is standard in the receiver operating characteristic (ROC) literature. Some basic results on this topic are recalled in Appendix B in relation to the above assumption.*

By the following result, the optimal classifiers for problems (1) and (2) are the same. Furthermore, one direction of this equivalence also holds in an approximate sense. In particular, approximate solutions to $X \sim P_0$ vs. $X \sim P_X$ translate to approximate solutions for $X \sim P_0$ vs. $X \sim P_1$. The following theorem constitutes our main *learning reduction* in the sense of Beygelzimer et al. (2005):

Theorem 1 *Assume property* **(A)** *is satisfied. Consider any α, $0 \le \alpha \le 1$, and assume $\pi > 0$. Then for any $f \in \mathcal{F}$ the two following statements are equivalent:*

(i) $R_X(f) = R^*_{X,\alpha}(\mathcal{F})$ and $R_0(f) \le \alpha$.

(ii) $R_1(f) = R^*_{1,\alpha}(\mathcal{F})$ and $R_0(f) = \alpha$.

More generally, let $L_{1,\alpha}(f,\mathcal{F}) = R_1(f) - R^*_{1,\alpha}(\mathcal{F})$ and $L_{X,\alpha}(f,\mathcal{F}) = R_X(f) - R^*_{X,\alpha}(\mathcal{F})$ denote the excess losses (regrets) for the two problems, and assume $\pi > 0$. If $R_0(f) \le \alpha + \varepsilon$ for $\varepsilon \ge 0$, then

$$L_{1,\alpha}(f,\mathcal{F}) \le \pi^{-1}(L_{X,\alpha}(f,\mathcal{F}) + (1-\pi)\varepsilon).$$

Proof . For any classifier f, we have the relation $R_X(f) = (1-\pi)(1-R_0(f)) + \pi R_1(f)$. We start with proving $(ii) \Rightarrow (i)$. Consider $f \in \mathcal{F}$ such that $R_1(f) = R^*_{1,\alpha}(\mathcal{F})$ and $R_0(f) = \alpha$, but assume $R_X(f) > R^*_X(\mathcal{F})$. Let $f' \in \mathcal{F}$ such that $R_X(f') < R_X(f)$ and $R_0(f') \le \alpha$. Then since $\pi > 0$,

$$\begin{aligned} R_1(f') &= \pi^{-1}\left(R_X(f') - (1-\pi)(1-R_0(f'))\right) \\ &< \pi^{-1}\left(R_X(f) - (1-\pi)(1-\alpha)\right) \\ &= R_1(f), \end{aligned}$$

contradicting minimality of $R_1(f)$.

To establish the converse implication, consider $f \in \mathcal{F}$ such that $R_X(f) = R^*_{X,\alpha}(\mathcal{F})$ and $R_0(f) \leq \alpha$, but assume $R_1(f) > R^*_{1,\alpha}(\mathcal{F})$ or $R_0(f) < \alpha$. Let f' be such that $R_0(f') = \alpha$ and $R_1(f') = R^*_1(\mathcal{F})$, whose existence is ensured by assumption **(A)**. Then

$$\begin{aligned} R_X(f') &= (1-\pi)(1-\alpha) + \pi R_1(f') \\ &< (1-\pi)(1-R_0(f)) + \pi R_1(f) \\ &= R_X(f), \end{aligned}$$

contradicting minimality of $R_X(f)$. To prove the final statement, first note that we established $R^*_{X,\alpha}(\mathcal{F}) = \pi R^*_{1,\alpha}(\mathcal{F}) + (1-\pi)(1-\alpha)$, by the first part of the theorem. By subtraction we have

$$\begin{aligned} L_{1,\alpha}(f,\mathcal{F}) &= \pi^{-1}(L_{X,\alpha}(f,\mathcal{F}) + (1-\pi)(R_0(f)-\alpha)) \\ &\leq \pi^{-1}(L_{X,\alpha}(f,\mathcal{F}) + (1-\pi)\varepsilon). \end{aligned}$$

■

Theorem 1 suggests that we may estimate the solution to (1) by solving a surrogate binary classification problem, treating $x_1, \ldots, x_m$ as one class and $x_{m+1}, \ldots, x_{m+n}$ as the other.

In the rest of the paper, we explore the consequences of this reduction from a theoretical as well as practical perspective. In the next section, we illustrate on the theoretical side, in the case of an empirical risk minimization (ERM) type algorithm, how a finite sample bound for NP classification translates to a finite sample bound for SSND and leads to desirable properties such as consistency. On the other hand, algorithms we can analyze (such as ERM) often do not have the best performance on actual data, and may be computationally infeasible (a situation that is not specific to SSND). Thus in the experimental Section 8 we implement a different method, namely simple but effective schemes based on kernel density estimates. It is important to observe that Theorem 1 still applies to these methods since it just compares two objective functions and is agnostic to the method used.

4. Statistical Performance Guarantees

We now illustrate how Theorem 1 leads to performance guarantees for SSND. We consider the case of a fixed set of classifiers $\mathcal{F}$ having finite VC-dimension (Vapnik, 1998), and the NP classification algorithm

$$\begin{aligned} \widehat{f}_\tau = \arg\min_{f \in \mathcal{F}} & \widehat{R}_X(f) \\ \text{s.t. } & \widehat{R}_0(f) \leq \alpha + \tau, \end{aligned}$$

based on (constrained) empirical risk minimization, where

$$\widehat{R}_X(f) = \frac{1}{n} \sum_{i=m+1}^{m+n} \mathbf{1}_{\{f(x_i) \neq 1\}}, \quad \widehat{R}_0(f) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}_{\{f(x_i) \neq 0\}}.$$

This rule was analyzed in Cannon et al. (2002) and Scott and Nowak (2005). Define the precision of a classifier f for class i as $Q_i(f) = P_{XY}(Y = i | f(X) = i)$ (the higher the precision, the better the

performance). Then we have the following result bounding the difference of the quantities R_i and Q_i to their optimal values over $\mathcal{F}$:

Theorem 2 *Assume $x_1,\ldots,x_m$ and $x_{m+1},\ldots,x_{m+n}$ are i.i.d. realizations of P_0 and P_X, respectively, and that the two samples are independent of each other. Assume $\pi > 0$. Let $\mathcal{F}$ be a set of classifiers of VC-dimension V. Assume property* **(A)** *is satisfied and denote by f^* an optimal classifier in $\mathcal{F}$ with respect to the criterion in* (1)*. Fixing $\delta > 0$, define $\varepsilon_k = \sqrt{\frac{V\log k - \log\delta}{k}}$. There exist absolute constants c, c' such that, if we choose $\tau = c\varepsilon_n$, the following bounds hold with probability $1-\delta$:*

$$R_0(\widehat{f}_\tau) - \alpha \leq c'\varepsilon_n\,; \tag{3}$$

$$R_1(\widehat{f}_\tau) - R_1(f^*) \leq c'\pi^{-1}(\varepsilon_n + \varepsilon_m)\,; \tag{4}$$

$$Q_i(f^*) - Q_i(\widehat{f}_\tau) \leq \frac{c'}{P(f^*(X)=i)}(\varepsilon_n+\varepsilon_m),\ i=0,1\,. \tag{5}$$

The proof is given in Appendix A. The primary technical ingredients in the proof are Theorem 3 of Scott and Nowak (2005) and the learning reduction of Theorem 1 above. The above theorem shows that the procedure is consistent inside the class $\mathcal{F}$ for all criteria considered, that is, these quantities decrease (resp. increase) asymptotically to their value at f^*. This is in contrast to the statistical learning bounds previously obtained (Liu et al., 2002, Thm. 2), which do not imply consistency.

Following Scott and Nowak (2005), by extending suitably the argument and the method in the spirit of structural risk minimization over a sequence of classes $\mathcal{F}_k$ having the universal approximation property, we can conclude that this method is universally consistent (that is, relevant quantities converge to their value at f^*, where f^* is the solution of (1) over the set of all possible classifiers). Therefore, although technically simple, the reduction result of Theorem 1 allows us to deduce stronger results than the existing ones concerning this problem. This can be paralleled with the result that inductive novelty detection can be reduced to classification against uniform data (Steinwart et al., 2005), which made the statistical learning study of that problem significantly simpler.

It is interesting to note that the multiplicative constant in front of the rate of convergence of the precision criteria is $P_X(f^*(X)=i)^{-1}$ rather than π^{-1} for R_1. In particular $P_X(f^*(X)=0) \geq (1-\pi)(1-\alpha)$, so that the convergence rate for class 0 precision is not significantly affected as $\pi \to 0$. Similarly $P_X(f^*(X)=1) \geq (1-\pi)\alpha$, so the convergence rate for class 1 precision depends more crucially on the (known) α than on π.

For completeness, we briefly discuss the optimality of $Q_i(f^*)$ in (5) in the sense of the criterion Q_i itself. Under an additional minor condition, it is possible to show (the details are given at the end of Appendix B) that under the constraint $R_0(f) \leq \alpha$, the best attainable precision for class 0 in the set $\mathcal{F}$ is attained by $f = f^*$. Therefore, in (5) ($i=0$), we are really comparing the precision of $\widehat{f}_\tau$ against the best possible class 0 precision given the FPR constraint. On the other hand, it does not make sense to consider the best attainable class 1 precision under an upper constraint on R_0, since we can have both $R_0 \to 0$ and $Q_1 \to 1$ by only rejecting a vanishingly small proportion of very sure novelties. But it can easily be seen that f^* realizes the best attainable class 1 precision under the *equality* constraint $R_0(f) = \alpha$.

We emphasize that the above result is but one of many possible theorems that could be deduced from the learning reduction; other results from Neyman-Pearson classification could also be applied. We also remark that, although the previous theorem corresponds to the semi-supervised setting, an

analogous transductive result is easily obtained by incorporating an additional uniform deviation bound relating the empirical error rates on the unlabeled data to the true error rates.

5. The Case $\pi = 0$ and a Hybrid Method

The preceding reduction of SSND to NP classification is only justified when $\pi > 0$. Aside from the analysis breaking down, this can be seen as follows. The unlabeled sample is a draw from $P_X = (1-\pi)P_0 + \pi P_1$. When $\pi = 0$, the unlabeled sample is a draw from P_0. Therefore it contains no information about P_1. Were we to solve the surrogate NP problem, we would be attempting to classify between two identical distributions, and the best we could do would be random guessing. This is confirmed in Table 2 (case $\pi = 0$) where the AUC values for SSND are near one half. Our goal in this section is to develop a learning reduction, and a parallel result to Theorem 1 in Section 3, but which handles the case $\pi = 0$ more sensibly.

When $\pi = 0$, we have no information about P_1 in either sample. Therefore, the only way to get any traction on the problem is to make some assumption about P_1. The inductive method makes such an assumption (as noted previously in the paper), namely, that P_1 is uniform on the support of P_0. Since uniformity is the standard assumption without any additional prior knowledge, we aim to develop a method that performs at least as well as the inductive method when $\pi = 0$.

Therefore we ask the following question: Can we devise a method which, having no knowledge of π, shares the properties of the learning reduction of Section 3 when $\pi > 0$, and the inductive approach otherwise? Our answer to the question is "yes" under fairly general conditions.

The intuition behind our approach is the following. The inductive approach to novelty detection performs density level set estimation. Furthermore, as we saw in Section 3, density level sets are optimal decision regions for testing the nominal distribution against a uniform distribution. Therefore, level set estimation can be achieved by generating an artificial uniform sample and performing weighted binary classification against the nominal data (this idea has been developed in more detail by Steinwart et al., 2005). Our approach is to sprinkle a vanishingly small proportion of uniformly distributed data among the unlabeled data, and then implement SSND using NP classification on this modified data. When $\pi = 0$, the uniform points will influence the final decision rule to perform level set estimation. When $\pi > 0$, the uniform points will be swamped by the actual novelties, and the optimal detector will be estimated.

To formalize this approach, let $0 < p_n < 1$ be a sequence tending to zero. Assume that S is a compact set which is known to contain the support of P_0 (obtained, e.g., through support estimation or through a priori information on the problem), and let P_2 be the uniform distribution on S. Consider the following procedure: Let $k \sim \text{binom}(n, p_n)$. Draw k independent realizations from P_2, and redefine $x_{m+1}, \ldots, x_{m+k}$ to be these values. (In practice, the uniform data would simply be appended to the unlabeled data, so that information is not erased. The present procedure, however, is slightly simpler to analyze.)

The idea now is to apply the SSND learning reduction from before to this modified unlabeled data. Toward this end, we introduce the following notations. For simplicity, we do not explicitly indicate the underlying class $\mathcal{F}$. We refer to any data point that was drawn from either P_1 or P_2 as an *operative* novelty. The proportion of operative novelties in the modified unlabeled sample is $\tilde{\pi} := \pi(1-p_n) + p_n$. The distribution of operative novelties is $\tilde{P}_1 := \frac{\pi(1-p_n)}{\tilde{\pi}} P_1 + \frac{p_n}{\tilde{\pi}} P_2$, and the overall distribution of the modified unlabeled data is $\tilde{P}_X := \tilde{\pi}\tilde{P}_1 + (1-\tilde{\pi})P_0$. Let $R_2, R^*_{2,\alpha}, \tilde{R}_1, \tilde{R}^*_{1,\alpha}, \tilde{R}_X$, and

$\tilde{R}^*_{X,\alpha}$ be defined in terms of $P_2, \tilde{P}_1$, and $\tilde{P}_X$, respectively, in analogy to the definitions in Section 3. Also denote $L_{2,\alpha}(f) = R_2(f) - R^*_{2,\alpha}$, $\tilde{L}_{1,\alpha}(f) = \tilde{R}_1(f) - \tilde{R}^*_{1,\alpha}$, and $\tilde{L}_{X,\alpha} = \tilde{R}_X(f) - \tilde{R}^*_{X,\alpha}$.

By applying Theorem 1 to the modified data, we immediately conclude that if $R_0(f) \leq \alpha + \varepsilon$, then

$$\tilde{L}_{1,\alpha}(f) \leq \frac{1}{\tilde{\pi}}(\tilde{L}_{X,\alpha}(f) + (1-\tilde{\pi})\varepsilon) = \frac{1}{\tilde{\pi}}(\tilde{L}_{X,\alpha}(f) + (1-\pi)(1-p_n)\varepsilon). \tag{6}$$

By previously cited results on Neyman-Pearson classification, the quantities on the right-hand side can be made arbitrarily small as m and n grow. The following result translates this bound to the kind of guarantee we are seeking.

Theorem 3 *Assume* **(A)** *holds. Let f be a classifier with $R_0(f) \leq \alpha + \varepsilon$. If $\pi = 0$, then*

$$L_{2,\alpha}(f) \leq p_n^{-1}(\tilde{L}_{X,\alpha}(f) + (1-p_n)\varepsilon).$$

If $\pi > 0$, then

$$L_{1,\alpha}(f) \leq \frac{1}{\pi(1-p_n)}(\tilde{L}_{X,\alpha}(f) + (1-\pi)(1-p_n)\varepsilon + p_n).$$

To interpret the first statement, note that $L_{2,\alpha}(f)$ is the inductive regret. The bound implies that $L_{2,\alpha}(f) \to 0$ as long as both $\varepsilon = R_0(f) - \alpha$ and $\tilde{L}_{X,\alpha}(f)$ tend to zero *faster than* p_n. This suggests taking p_n to be a sequence tending to zero slowly. The second statement is similar to the earlier result in Theorem 1, but with additional factors of p_n. These factors suggest choosing p_n tending to zero rapidly, in contrast to the first statement, so in practice some balance should be struck.

Proof If $\pi = 0$, then $\tilde{L}_{1,\alpha} = L_{2,\alpha}$ and the first statement follows trivially from (6). To prove the second statement, denote $\beta_n := \frac{\pi(1-p_n)}{\tilde{\pi}}$, and observe that

$$\begin{aligned}
\tilde{R}^*_{1,\alpha} &= \inf_{R_0(f)\leq\alpha} \tilde{R}_1(f) \\
&= \inf_{R_0(f)\leq\alpha} [\beta_n R_1(f) + (1-\beta_n)R_2(f)] \\
&\leq \beta_n R^*_{1,\alpha} + (1-\beta_n).
\end{aligned}$$

Therefore

$$\begin{aligned}
\tilde{L}_{1,\alpha}(f) &= \tilde{R}_1(f) - \tilde{R}^*_{1,\alpha} \\
&\geq \beta_n R_1(f) + (1-\beta_n)R_2(f) - \beta_n R^*_{1,\alpha} - (1-\beta_n) \\
&\geq \beta_n(R_1(f) - R^*_{1,\alpha}) - (1-\beta_n) \\
&= \beta_n L_{1,\alpha}(f) - (1-\beta_n)
\end{aligned}$$

and we conclude, still using (6),

$$\begin{aligned}
L_{1,\alpha}(f) &\leq \frac{1}{\beta_n}\tilde{L}_{1,\alpha} + \frac{1-\beta_n}{\beta_n} \\
&\leq \frac{1}{\pi(1-p_n)}(\tilde{L}_{X,\alpha}(f) + (1-\pi)(1-p_n)\varepsilon + p_n).
\end{aligned}$$

■

Like Theorem 1, Theorem 3 is quite general, and has both theoretical and practical implications. Theoretically, it could be combined with specific, analyzable algorithms for Neyman-Pearson classification to yield novelty detectors with performance guarantees, as was illustrated in Section 4. We do not develop this theoretical direction here. Practically, any algorithm for Neyman-Pearson classification that generally works well in practice can be applied in the hybrid framework to produce novelty detectors that perform well for values of π that are zero or very near zero. We implement this idea in the experimental section below.

We also remark that this hybrid procedure could be applied with any prior distribution on novelties besides uniform. In addition, the hybrid approach could also be practically useful when n is small, assuming the artificial points are appended to the unlabeled sample.

6. Estimating π and Testing for $\pi = 0$

In the previous sections, our main goal was to find a good classifier function for the purpose of novelty detection. Besides the detector itself, it is often relevant to the user to have an estimate or bound on the proportion π of novelties in the contaminated distribution P_X. Estimation of π allows for estimating and optimizing the misclassification rate on the unlabeled data, which is often of interest in the LPUE literature (see Sec. 2). Estimation of π is also useful for estimating the precision (as defined in Section 4); this topic will be revisited in the next section in the context of multiple testing.

It may also be useful to test whether there are novelties at all; in other words, since the learnt detector $\widehat{f}$ is allowed a certain proportion of false positives, it is important to assess whether the reported novelties are a statistically significant indication of the presence of true novelties, or if they are likely to be all false positives. We focus on these issues in the present section.

It should first be noted that without additional assumptions, π is not an identifiable parameter in our model. To see this, consider the idealized case where we have an infinite amount of nominal and contaminated data, so that we have perfect knowledge of P_0 and P_X. Assuming the decomposition $P_X = (1-\pi)P_0 + \pi P_1$ holds, note that any alternate decomposition of the form $P_X = (1-\pi-\gamma)P_0 + (\pi+\gamma)P_1'$, with $P_1' = (\pi+\gamma)^{-1}(\pi P_1 + \gamma P_0)$, and $\gamma \in [0, 1-\pi]$, is equally valid. Because the most important feature of the model is that we have no direct knowledge of P_1, we cannot decide which representation is the "correct" one; we could not even exclude *a priori* the case where $\pi = 1$ and $P_1 = P_X$ (while producing the exact same observed data). The previous results established in Theorems 1-3 are valid for whatever underlying representation is assumed to be correct. For the estimation of the proportion of novelties, however, it makes sense to define π as the *minimal* proportion of novelties that can explain the difference between P_0 and P_X. First we introduce the following definition:

Definition 4 *Assume P_0, P_1 are probability distributions on the measure space $(\mathcal{X}, \mathfrak{S})$. We call P_1 a* proper novelty distribution *with respect to P_0 if there exists no decomposition of the form $P_1 = (1-\gamma)Q + \gamma P_0$ where Q is some probability distribution and $0 < \gamma \leq 1$.*

This defines a proper novelty distribution P_1 as one that cannot be confounded with P_0; it cannot be represented as a (nontrivial) mixture of P_0 with another distribution.

The next result establishes a canonical decomposition of the contaminated distribution into a mixture of nominal data and proper novelties. As a consequence the proportion π^* of proper novelties, and therefore the proper novelty distribution P_1 itself, are well-defined (that is, identifiable)

given the knowledge of the nominal and contaminated distributions (except for the special case $P_0 = P_X$, where of course the novelty distribution is not defined).

Proposition 5 *Assume P_0 , P_X are probability distributions on the measure space $(\mathcal{X}, \mathfrak{S})$. If $P_X \neq P_0$, there is a unique $\pi^* \in (0,1]$ and P_1 such that the decomposition $P_X = (1-\pi^*)P_0 + \pi^* P_1$ holds, and such that P_1 is a proper novelty distribution with respect to P_0 . If we additionally define $\pi^* = 0$ when $P_X = P_0$, then in all cases,*

$$\pi^* := \min\{\alpha \in [0,1] : \exists Q \text{ probability distribution: } P_X = (1-\alpha)P_0 + \alpha Q\} . \tag{7}$$

The proof is given in Appendix A. For the rest of this section we assume for simplicity of notation that π and P_1 are the proportion and distribution of proper novelties of P_X with respect to P_0 . The results to come are also informative for improper novelty distributions, in the following sense: if P_1 is not a proper novelty distribution and the decomposition $P_X = (1-\pi)P_0 + \pi P_1$ holds, then (7) entails that $\pi > \pi^*$. It follows that a lower bound on π^* (either deterministic or valid with a certain confidence), as will be derived in the coming sections, is always also a valid lower confidence bound on π when non-proper novelties are considered. A lower bound is effectively the best we can hope for π if P_1 is not assumed to be proper.

6.1 Population Case

We now want to relate the estimation of π to quantities previously introduced and problem (1). We first treat the population case and optimal novelty detection over the set of all possible classifiers.

Theorem 6 *For any classifier f, we have the inequality*

$$\pi \geq 1 - \frac{R_X(f)}{1 - R_0(f)} .$$

Optimizing this bound over a set of classifiers $\mathcal{F}$ under the FPR constraint $R_0(f) \leq \alpha$ yields for any $0 \leq \alpha < 1$:

$$\pi \geq 1 - \frac{R^*_{X,\alpha}(\mathcal{F})}{1-\alpha} . \tag{8}$$

Furthermore, if $\mathcal{F}$ is the set of all deterministic classifiers,

$$\pi = 1 - \inf_{\alpha \in [0,1)} \frac{R^*_{X,\alpha}(\mathcal{F})}{1-\alpha} . \tag{9}$$

Proof . For the first part, just write for any classifier f

$$\begin{aligned} 1 - R_X(f) &= P_X(f(X) = 1) \\ &= (1-\pi)P_0(f(X) = 1) + \pi P_1(f(X) = 1) \\ &\leq (1-\pi)R_0(f) + \pi , \end{aligned}$$

resulting in the first inequality in the theorem. Under the constraint $R_0(f) \leq \alpha$, this inequality then yields

$$\pi \geq 1 - \frac{R_X(f)}{1 - R_0(f)} \geq 1 - \frac{R_X(f)}{1-\alpha} ;$$

optimizing the bound under the constraint yields the second inequality.

We establish in Lemma 13 in Appendix A that for any $\varepsilon > 0$ there exists a classifier f_ε such that $R_0(f_\varepsilon) < 1$ and $R_1(f_\varepsilon)/(1-R_0(f_\varepsilon)) \leq \varepsilon$. Put $\alpha_\varepsilon = R_0(f_\varepsilon)$; we then have

$$R^*_{X,\alpha_\varepsilon}(\mathcal{F}) \leq R_X(f_\varepsilon) = (1-\pi)(1-\alpha_\varepsilon) + \pi R_1(f_\varepsilon)\,,$$

implying

$$\pi \geq 1 - \inf_{\alpha\in[0,1)} \frac{R^*_{X,\alpha}(\mathcal{F})}{1-\alpha} \geq 1 - \frac{R^*_{X,\alpha_\varepsilon}(\mathcal{F})}{1-\alpha_\varepsilon} \geq \pi\left(1 - \frac{R_1(f_\varepsilon)}{1-R_0(f_\varepsilon)}\right) \geq \pi(1-\varepsilon)\,,$$

which establishes the last claim of the theorem. ■

6.2 Distribution-free Lower Confidence Bounds on π

In the last part of Theorem 6, if we assume that the function $\alpha \mapsto R^*_{X,\alpha}(\mathcal{F})/(1-\alpha)$ is nonincreasing (a common regularity assumption; see Appendix B for a discussion of how this condition can always be ensured by considering possibly randomized classifiers), then $\alpha \mapsto R^*_{X,\alpha}(\mathcal{F})$ is left differentiable at $\alpha = 1$ and (8) is optimized by taking $\alpha \to 1$, that is,

$$\pi \geq 1 + \frac{dR^*_{X,\alpha}(\mathcal{F})}{d\alpha}\Big|_{\alpha=1^-}, \tag{10}$$

while (9) entails that the above inequality is an equality if $\mathcal{F}$ contains all deterministic classifiers. This suggests obtaining a lower bound on π by estimating the slope of $R^*_{X,\alpha}(\mathcal{F})$ at its right endpoint. The following result adopts this approach while accounting for the uncertainty inherent in empirical performance measures.

Theorem 7 *Consider a classifier set $\mathcal{F}$ for which we assume a uniform error bound of the following form is available: for any distribution Q on $\mathcal{X}$, with probability at least $1-\delta$ over the draw of an i.i.d. sample of size n according to Q , we have*

$$\forall f \in \mathcal{F} \quad \left|Q(f(X)=1) - \widehat{Q}(f(X)=1)\right| \leq \varepsilon_n(\mathcal{F},\delta)\,, \tag{11}$$

where $\widehat{Q}$ denotes the empirical distribution built on the sample.

Then the following quantity is a lower bound on π with probability at least $(1-\delta)^2 \geq 1-2\delta$ (over the draw of the nominal and unlabeled samples) :

$$\widehat{\pi}^-(\mathcal{F},\delta) := 1 - \inf_{f\in\mathcal{F}} \frac{\widehat{R}_X(f) + \varepsilon_n(\mathcal{F},\delta)}{(1-\widehat{R}_0(f) - \varepsilon_m(\mathcal{F},\delta))_+}\,, \tag{12}$$

where the ratio is formally defined to be 1 whenever the denominator is 0 .

Note that if we define $\widehat{f}_\alpha = \arg\min_{f\in\mathcal{F}} \widehat{R}_X(f)$ under the constraint $\widehat{R}_0(f) \leq \alpha$, this can be rewritten

$$\widehat{\pi}^-(\mathcal{F},\delta) = 1 - \inf_{\alpha\in[0,1]} \frac{\widehat{R}_X(\widehat{f}_\alpha) + \varepsilon_n(\mathcal{F},\delta)}{(1-\widehat{R}_0(\widehat{f}_\alpha) - \varepsilon_m(\mathcal{F},\delta))_+}\,.$$

There are two balancing forces at play here. From the population version (10) (valid under a mild regularity assumption), we know that we would like to have α as close as possible to 1 for estimating the derivative of $R^*_{X,\alpha}(\mathcal{F})$ at $\alpha = 1$. This is balanced by the estimation error which makes estimations close to $\alpha = 1$ unreliable because of the denominator. Taking the infimum along the curve takes in a sense the best available bias-estimation tradeoff.

Proof . To simplify notation we denote $\varepsilon_n(\mathcal{F}, \delta)$ simply by ε_n. As in the proof of the previous result, write for any classifier f:

$$P_X(f(X) = 1) \leq (1 - \pi) P_0(f(X) = 1) + \pi,$$

from which we deduce after applying the uniform bound

$$\begin{aligned} 1 - \widehat{R}_X(f) - \varepsilon_n &= \widehat{P}_X(f(X) = 1) - \varepsilon_n \\ &\leq (1 - \pi)(\widehat{R}_0(f) + \varepsilon_m) + \pi, \end{aligned}$$

which can be solved whenever $1 - \widehat{R}_0(f) - \varepsilon_m > 0$. ■

The following result shows that $\widehat{\pi}^-(\mathcal{F}, \delta)$, when suitably applied using a sequence of classifier sets $\mathcal{F}_1, \mathcal{F}_2, \ldots$ that have a universal approximation property, yields a strongly universally consistent estimate of the proportion π of proper novelties. The proof is given in Appendix A and relies on Theorem 7 in conjunction with the Borel-Cantelli lemma.

Theorem 8 *Consider a sequence $\mathcal{F}_1, \mathcal{F}_2, \ldots$ of classifier sets having the following universal approximation property: for any measurable function $f^* : \mathcal{X} \to \{0, 1\}$, and any distribution Q , we have*

$$\liminf_{k \to \infty} \inf_{f \in \mathcal{F}_k} Q(f(X) \neq f^*(X)) = 0 .$$

Suppose also that each class $\mathcal{F}_k$ has finite VC-dimension V_k, so that for each $\mathcal{F}_k$ we have a uniform confidence bound of the form (11) *for* $\varepsilon_n(\mathcal{F}_k, \delta) = 3\sqrt{\frac{V_k \log(n+1) - \log \delta/2}{n}}$ *. Define*

$$\widehat{\pi}^-(\delta) = \sup_k \widehat{\pi}^- \left(\mathcal{F}_k, \delta k^{-2}\right) .$$

If $\delta = (mn)^{-2}$, *then* $\widehat{\pi}^-$ *converges to* π *almost surely as* $\min(m, n) \to \infty$.

6.3 There are No Distribution-free Upper Bounds on π

The lower confidence bounds $\widehat{\pi}^-(\mathcal{F}, \delta)$ and $\widehat{\pi}^-(\delta)$ are distribution-free in the sense that they hold regardless of P_0, P_1 and π. We now argue that distribution-free upper confidence bounds do not generally exist.

We define a *distribution-free upper confidence bound* $\widehat{\pi}^+(\delta)$ to be a function of the observed data such that, for any P_0, any proper novelty distribution P_1, and any novelty proportion $\pi \leq 1$, we have $\widehat{\pi}^+(\delta) \geq \pi$ with probability $1 - \delta$ over the draw of the two samples.

We will show that such a universal upper bound does not exist unless it is trivial. The reason is that the novel distribution can be arbitrarily hard to distinguish from the nominal distribution. It is possible to detect with some certainty that there is a non-zero proportion of novelties in the contaminated data (see Corollary 11 below), but we can never be sure that there are no novelties.

This situation is similar to the philosophy of significance testing: one can never accept the null hypothesis, but only have insufficient evidence to reject it.

We will say that the nominal distribution P_0 is *weakly diffuse* if for any $\gamma > 0$ there exists a set A such that $0 < P_0(A) < \gamma$. We say an upper confidence bound $\widehat{\pi}^+(\delta)$ is *non-trivial* if there exists a weakly diffuse nominal distribution P_0, a proper novelty distribution P_1, a novelty proportion $\pi < 1$, and a constant $\delta > 0$ such that

$$P(\widehat{\pi}^+(\delta) < 1) > \delta,$$

where the probability is over the joint draw of nominal and contaminated samples. This assumption demands that there is at least a specific setting where the upper bound $\widehat{\pi}^+(\delta)$ is significantly different from the trivial bound 1, meaning that it is bounded away from 1 with larger probability than its allowed probability of error δ.

Theorem 9 *There exists no distribution-free, non-trivial upper confidence bound on* π.

The proof appears in Appendix A. The non-triviality assumption is quite weak and relatively intuitive. The only not directly intuitive assumption is that P_0 should be weakly diffuse, which is satisfied for all distributions having a continuous part. This assumption effectively excludes finite state spaces, which is an important condition: if $\mathcal{X}$ is finite, it is actually possible to obtain a non-trivial upper confidence bound on π.

The following corollary establishes that for any finite sample size, any estimator of π (and in particular the universally consistent estimator considered in the previous section) can have an average error bounded from below by a constant independent of the sample size.

Corollary 10 *Assume $\mathcal{X}$ is an infinite set and let m,n be fixed. For any estimator $\widehat{\pi}$ of π, based on a joint sample of size (m,n), and any fixed real $p > 0$:*

$$\sup_{P \in \mathcal{P}(m,n)} E\left[|\widehat{\pi} - \pi|^p\right] \geq c(p) > 0,$$

where $\mathcal{P}(m,n)$ denotes the set of all generating distributions of (m,n)-samples following the SSND model (that is, of the form $P = P_0^{\otimes m} \otimes P_X^{\otimes n}$ for arbitrary P_0, P_X), and $c(p)$ is a constant independent of (m,n).

This result essentially precludes the existence of universal convergence rates in the estimation of π. In other words, to achieve some prescribed rate of convergence, some assumptions on the generating distributions must be made. This parallels the estimation of the Bayes risk in classification (Devroye, 1982).

6.4 Testing for $\pi = 0$

The lower confidence bound on π can also be used as a test for $\pi = 0$, that is, a test for whether there are any novelties in the test data:

Corollary 11 *Let $\mathcal{F}$ be a set of classifiers. If $\widehat{\pi}^-(\mathcal{F}, \delta) > 0$, then we may conclude, with confidence $1 - \delta$, that the unlabeled sample contains novelties.*

It is worth noting that testing this hypothesis is equivalent to testing if P_0 and P_X are the same distribution, which is the classical two-sample problem in an arbitrary input space. This problem has recently generated attention in the machine learning community (Gretton et al., 2007), and the approach proposed here, using arbitrary classifiers, seems to be new. Our confidence bound could of course also be used to test the more general hypothesis $\pi \leq \pi_0$ for a prescribed $\pi_0, 0 \leq \pi_0 < 1$.

Note that, by definition of $\widehat{\pi}^-(\mathcal{F},\delta)$, testing the hypothesis $\pi = 0$ using the above lower confidence bound for π is equivalent to searching the classifier space $\mathcal{F}$ for a classifier f such that the proportions of predictions of 0 and 1 by f differ on the two samples in a statistically significant manner. Namely, for a classifier f belonging to a class $\mathcal{F}$ for which we have a uniform bound of the form (11), we have the lower bound $P_X(f(X)=1) \geq \widehat{P}_X(f(X)=1) - \varepsilon_n$ and the upper bound $P_0(f(X)=1) \leq \widehat{P}_0(f(X)=1) + \varepsilon_m$ (both bounds valid simultaneously with probability at least $1-\delta$). If the difference of the bounds is positive we conclude that we must have $P_X \neq P_0$, hence $\pi > 0$. This difference is precisely what appears in the numerator of $\widehat{\pi}^-(\mathcal{F},\delta)$ in (12) . Furthermore, if this numerator is positive then so is the denominator, since it is always larger. In the end, $\widehat{\pi}^-(\mathcal{F},\delta) > 0$ is equivalent to

$$\sup_{f\in\mathcal{F}} \Big((\widehat{P}_X(f(X)=1) - \varepsilon_n) - (\widehat{P}_0(f(X)=1) + \varepsilon_m) \Big) > 0 \,.$$

7. Relationship Between SSND and Multiple Testing

In this section, we show how SSND offers powerful generalizations of the standard p-value approach to multiple testing under the widely used "random effects" model, as considered for example by Efron et al. (2001).

7.1 Multiple Testing Under the Random Effects Model

In the multiple testing framework, a finite family $(H_1,\ldots,H_K)$ of null hypotheses to test is fixed; from the observation of some data X , a decision $D(H_i,X) \in \{0,1\}$ must be taken for each hypothesis, namely whether (given the data) hypothesis H_i is deemed to be false ($D(H_i,X)=1$, hypothesis rejected) or true ($D(H_i,X)=0$, hypothesis not rejected). A typical application domain is that of microarray data analysis, where each null hypothesis H_i corresponds to the absence of a difference in expression levels of gene i in a comparison between two experimental situations. A rejected null hypothesis then indicates such a differential expression for a specific gene, and is called a *discovery* (since differentially expressed genes are those of interest). However, the number of null hypotheses to test is very large, for example $K \simeq 4.10^4$ in the gene expression analysis, and the probability of rejecting by chance a null hypothesis must be strictly controlled.

In the standard setting for multiple testing, it is assumed that a testing statistic $Z_i(X) \in \mathbb{R}$ has been fixed for each null hypothesis H_i , and that its marginal distribution is known when H_i is true. This statistic can then be normalized (by suitable monotone transform) to take the form of a *p-value*. A p-value is a function $p_i(X)$ of the data such that, if the corresponding null hypothesis H_i is true, then $p_i(X)$ has a uniform marginal distribution on $[0,1]$. In this setting, it is expected that the rejection decisions $D(H_i,X)$ are taken based on the observed p-values $(p_1(X),\ldots,p_K(X))$ rather than on the raw data. In fact, in most cases it is assumed that the decisions take the form $D(H_i,X) = \mathbf{1}_{\{p_i(X)\leq \widehat{T}\}}$, where $\widehat{T}$ is a data-dependent threshold. Further, simplifying distributional assumptions on the family of p-values are often posited. A common distribution model called

random effects abstracts the p-values from the original data X and assumes that the veracity of hypothesis H_i is governed by an underlying latent variable h_i as follows:

- the variables $h_i \in \{0,1\}$, $1 \leq i \leq K$ are i.i.d. Bernoulli with parameter π
- the variables p_i are independent, and conditionally to $(h_1, \ldots, h_K)$ have distribution

$$p_i \sim \begin{cases} \text{Uniform}[0,1]\,, & \text{if } h_i = 0 \\ P_1\,, & \text{if } h_i = 1\,. \end{cases}$$

Under the random effects model, the p-values thus follow a mixture distribution $(1-\pi)U[0,1] + \pi P_1$ on the interval $[0,1]$ and can be seen as a contaminated sample, while the variables h_i play the role of the unknown labels. It should now be clear that the above model is in fact a *specification* of the SSND model, with the following additional assumptions:

1. The observation space is the interval $[0,1]$;
2. The nominal distribution P_0 is known to be exactly uniform on $[0,1]$ (equivalently, the nominal distribution is uniform and the nominal sample has infinite size);
3. The class of novelty detectors considered is the set of intervals of the form $[0,t], t \in [0,1]$.

Therefore, the results developed in this paper can apply to the more restricted setting of multiple testing under the random effects model as well. In particular, the estimator $\widehat{\pi}^-(\mathcal{F}, \delta)$ developed in Section 6, when specified under the above additional conditions, recovers the methodology of non-asymptotic estimation of $1-\pi$ which was developed by Genovese and Wasserman (2004), Section 3, and our notion of proper novelty distribution recovers their notion of *purity* in that setting (and has somewhat more generality, since they assumed P_1 to have a density).

There are several interesting benefits in considering for the purpose of multiple testing the more general SSND model developed here. First, it can be unrealistic in practice to assume that the distribution of the p-values is known exactly under each one of the null hypotheses. Instead, only assuming the knowledge of a reference sample under controlled experimental conditions as in the SSND model is often more realistic. This problem was recently motivated by problems in genomics (Ghosh and Chinnaiyan, 2009) and proteomics (Ghosh, 2009), where in the latter reference asymptotic analysis was also presented.

Secondly, the restriction to decision sets of the form $\{p_i \leq t\}$ can also be questionable. For a single test, decision regions of this form are optimal (in the Neyman-Pearson sense) only if the likelihood ratio of the alternative to the null is decreasing, which amounts to assuming that the alternative distribution P_1 has a decreasing density. This assumption has been criticized in some recent work. A simple example of a situation where this assumption fails is in the framework of z or t-tests, that is, the null distribution of the statistic (before rescaling into p-values) is a standard Gaussian or a Student t-distribution, and the corresponding p-value function is the usual one- or two-sided p-value. If the alternative distribution P_1 is a mixture of Gaussians (resp. of noncentral t distributions), optimal rejection regions for the original statistic are in general a finite union of disjoint intervals and do not correspond to level sets of the p-values. In order to counter this type of problem, Sun and Cai (2007) suggest to estimate from the data the alternate density and the proportion of true null hypotheses, and use these estimates directly in a plug-in likelihood ratio

based test. Chi (2007) develops a procedure based on growing rejection intervals around a finite number of fixed control points in $[0,1]$. In both cases, an asymptotic theory is developed. Both of these procedures are more flexible than using only rejection intervals of the form $[0,t]$ and aim at adaptivity with respect to the alternative distribution P_1.

Finally, the remaining restriction that effective observations (the p-values) belong to the unit interval was also put into question by Chi (2008), who considered a setting of multidimensional p-values belonging to $[0,1]^d$. The distribution was still assumed to be uniform under the corresponding null hypothesis, although this seems an even less realistic assumption than in dimension one. In this framework, the use of a reference "nominal" sample under the null distribution seems even more relevant.

The framework developed in the present paper allows to cover at once these different types of extensions rather naturally by just considering a richer class $\mathcal{F}$ of candidate classifiers (or equivalently in this setting, rejection regions), and provides a non-asymptotical analysis of their behavior using classical learning theoretical tools such as VC inequalities. Furthermore, such non-asymptotic inequalities can also give rise to adaptive and consistent model selection for the set of classifiers using the structural risk minimization principle, a topic that was not addressed previously for the extensions mentioned above.

7.2 SSND with Controlled FDR

One remaining important difference between the SSND setting studied here and that of multiple testing is that our main optimization problem (1) is under a false positive rate constraint $R_0(f) \leq \alpha$, while most recent work on multiple testing generally imposes a constraint on the false discovery rate (FDR) instead. If we denote

$$\mathrm{Pos}(f) = \widehat{P}_X(f(X)=1) = 1 - \widehat{R}_X(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{f(x_{m+i})=1\}}$$

the proportion of reported novelties, and

$$\mathrm{FP}(f) = \widehat{P}_{XY}(f(X)=1, Y=0) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{f(x_{m+i})=1, y_{m+i}=0\}}$$

the (unavailable to the user) proportion of false discoveries on the contaminated sample, then the false discovery proportion (FDP) is defined as $\mathrm{FDP}(f) = \mathrm{FP}(f)/\mathrm{Pos}(f)$ (taken to be zero if the denominator vanishes), and the FDR is defined as $\mathrm{FDR}(f) = E[\mathrm{FDP}(f)]$. Some classical variations of this quantity are the positive FDR, $\mathrm{pFDR}(f) = E[\mathrm{FDP}(f)|\mathrm{Pos}(f) > 0]$ and the marginal FDR, $\mathrm{mFDR}(f) = E[\mathrm{FP}(f)]/E[\mathrm{Pos}(f)]$. Under the mixture contamination model, it can be checked that $\mathrm{pFDR}(f) = \mathrm{mFDR}(f) = P_{XY}(Y=0|f(X)=1)$ (Storey, 2003), hence also equal to one minus the precision for class 1 (as defined earlier in Section 4). The following result states explicit empirical bounds on these quantities:

Proposition 12 *Consider a classifier set $\mathcal{F}$ for which we assume a uniform error bound of the following form is available: for any distribution Q on $\mathcal{X} \times \{0,1\}$, with probability at least $1-\delta$ over the draw of an i.i.d. sample of size n according to Q, both*

$$\forall f \in \mathcal{F} \quad \left| Q(f(X)=1) - \widehat{Q}(f(X)=1) \right| \leq \varepsilon_n(\mathcal{F}, \delta), \tag{13}$$

and

$$\forall f \in \mathcal{F} \quad \left| Q(f(X)=1, Y=0) - \widehat{Q}(f(X)=1, Y=0) \right| \leq \varepsilon_n(\mathcal{F}, \delta), \tag{14}$$

hold, where $\widehat{Q}$ denotes the empirical distribution built on the sample.

Then the following inequalities hold with probability at least $(1-\delta)^2 \geq 1-2\delta$ (over the draw of the nominal and unlabeled samples) :

$$\forall f \in \mathcal{F} \;\; \text{mFDR}(f) = P_X(Y=0|X=1) \leq \frac{(\widehat{R}_0(f)+\varepsilon_m)(1-\widehat{\pi}^-(\mathcal{F},\delta))}{(1-\widehat{R}_X(f)-\varepsilon_n)_+},$$

and

$$\forall f \in \mathcal{F} \;\; \text{FDP}(f) \leq \frac{(\widehat{R}_0(f)+\varepsilon_m)(1-\widehat{\pi}^-(\mathcal{F},\delta))+\varepsilon_n}{(1-\widehat{R}_X(f))},$$

where $\widehat{\pi}^-(\mathcal{F},\delta)$ is defined in (12).

Note that Equations (13) and (14) hold as before with $\varepsilon_n(\mathcal{F},\delta) = c\sqrt{\frac{V\log n - \log\delta}{n}}$ when $\mathcal{F}$ has VC dimension V. In the interest of simplicity, we use the same bound ε_n for both uniform error assumptions. Separate bounds could also be adopted, allowing (13) to be slightly tighter. We also remark that since FDP is an empirical quantity based on the contaminated sample, the second bound is in fact a *transductive* bound rather than semi-supervised.

Proof . The mFDR can be rewritten as $\text{mFDR}(f) = P_0(f(X)=1|Y=0)P_{XY}(Y=0)/P_X(f(X)=1) = R_0(f)(1-\pi)/(1-R_X(f))$. In this expression we can plug in the lower bound for π of Theorem 7 and uniform bounds for $R_0(f)$ and $R_X(f)$ coming from assumption (13). The FDP can be written as $\text{FDP}(f) = \widehat{P}_{XY}(f(X)=1, Y=0)/(1-\widehat{R}_X(f))$. Using assumption (14), the numerator can be upper bounded by $P_{XY}(f(X)=1,Y=0)+\varepsilon_n = R_0(f)(1-\pi)+\varepsilon_n$, and we can then use the same reasoning as for the first part. ■

Similarly to what was proposed in Section 4 under the false positive rate constraint, we can in this context consider to maximize $\widehat{R}_X(f)$ over $f \in \mathcal{F}$ subject to the constraint that the above empirical bound on the mFDR or FDP is less than α. This can then be suitably extended to a sequence of classes $\mathcal{F}_k$. While a full study of the resulting procedure is out of the scope of the present paper, we want to point out the important difference that the mFDR is necessarily lower bounded by $\inf_{x\in\mathcal{X}} P_{XY}(Y=0|X=x)$ which is generally strictly positive. Hence, the required constraint may not be realizable if α is smaller than this lower bound, in which case the empirical procedure should return a failure statement with probability one as $n \to \infty$.

8. Experiments

Despite previous work on learning with positive and unlabeled examples (LPUE), as discussed in Section 2, the efficacy of our proposed learning reduction, compared to the method of inductive novelty detection, has not been empirically demonstrated. In addition, we evaluate our proposed hybrid method. To assess the impact of unlabeled data on novelty detection, we applied our framework to some data sets which are common benchmarks for binary classification. The first 13 data sets (Müller et al., 2001) are from `http://www.fml.tuebingen.mpg.de/Members/`

Data Set	dim	N_{train}	N_{test}	π_{base}
banana	2	400	4900	0.45
breast-cancer	9	200	77	0.29
diabetes	8	468	300	0.35
flare-solar	9	666	400	0.55
german	20	700	300	0.30
heart	13	170	100	0.44
ringnorm	20	400	7000	0.50
thyroid	5	140	75	0.30
titanic	3	150	2051	0.32
twonorm	20	400	7000	0.50
waveform	21	400	4600	0.33
image	18	1300	1010	0.57
splice	60	1000	2175	0.48
ionosphere	34	251	100	0.64
mushrooms	112	4124	4000	0.48
sonar	60	108	100	0.47
adult	123	3000	3000	0.24
web	300	3000	3000	0.03

Table 1: Description of data sets. dim is the number of features, and N_{train} and N_{test} are the numbers of training and test examples. π_{base} is the proportion of positive examples (novelties) in the combined training and test data. Thus, the average (across permutations) nominal sample size m is $(1-\pi_{\text{base}})N_{\text{train}}$.

`raetsch/benchmark` and the last five data sets (Chang and Lin, 2001) are from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

Each data set consists of both positive and negative examples. Furthermore, each data set is replicated 100 times (except for image and splice, which are replicated 20 times), with each copy corresponding to a different random partitioning into training and test examples. All numerical results for a data set were obtained by averaging across all partitions. The negative examples from the training set were taken to form the nominal sample, and the positive training examples were not used at all in the experiments. The data sets are summarized in Table 1. Here N_{train} and N_{test} are the sizes[1] of the training and test sets, respectively, while π_{base} is the proportion of positive examples in the combined training and test data. Thus, the average (across permutations) nominal sample size m is $(1-\pi_{\text{base}})N_{\text{train}}$.

We emphasize that in these experiments we do not implement the empirical risk minimization (ERM) algorithm from Sec. 4. The reduction to Neyman-Pearson classification is general and can by applied in conjunction with any NP classification algorithm, whether that algorithm has associated performance guarantees or not. We here elect to apply the reduction using a plug-in kernel density estimate (KDE) classifier. ERM is computationally infeasible, and the bounds tend

1. The web and adult data sets were subsampled owing to their large size.

to be too loose in practice to be effective. The KDE plug-in rule can be implemented efficiently, and there is a natural inductive counterpart for comparison, the thresholded KDE based on the nominal sample.

8.1 Experimental Setup

We evaluated our methodology in two learning paradigms, comparing five learning methods across several values of π. The two learning paradigms are semi-supervised and transductive. For semi-supervised learning, the test data were divided into two halves. The first half was used as the contaminated, unlabeled data. The second half was used as an independent sample of contaminated data, not used in the learning stage, but only for evaluation of classifiers returned by each method. In particular, the second half of the test data was used to estimate the area under the ROC (AUC) of each method. Here, the ROC is the one which views P_0 as the null distribution and P_1 as the alternative. For transductive learning, the entire test set was treated as the unlabeled data, and was also used for evaluating the AUC.

The learning methods are the inductive approach, our proposed learning reduction, and three versions of the hybrid approach. The three hybrids correspond to $p_n = 1.0, 0.5$, and 0.1, in which a uniform sample of size $100p_n\%$ of the unlabeled sample size is *appended* to the unlabeled data. We emphasize that each algorithm was implemented in the same way in the two learning paradigms; the only differences are the size of the contaminated sample, and how they are evaluated.

We implemented the inductive novelty detector using a thresholded kernel density estimate (KDE) with Gaussian kernel, and SSND using a plug-in KDE classifier. To alleviate concerns that our inductive implementation is inadequate, we also tested the one-class support vector machine (Schölkopf et al., 2001) in several experimental settings, and found its performance to be very similar. Letting k_σ denote a Gaussian kernel with bandwidth σ, the inductive novelty detector at density level λ is

$$f(x) = \begin{cases} 1 & \text{if } \frac{1}{m}\sum_{i=1}^{m} k_{\sigma_0}(x,x_i) > \lambda \\ 0 & \text{otherwise,} \end{cases}$$

and the SSND classifier at density ratio λ is

$$f(x) = \begin{cases} 1 & \text{if } (\frac{1}{n}\sum_{i=m+1}^{m+n} k_{\sigma_X}(x,x_i))/(\frac{1}{m}\sum_{i=1}^{m} k_{\sigma_0}(x,x_i)) > \lambda \\ 0 & \text{otherwise.} \end{cases}$$

The hybrid method is implemented similarly. The ROCs of these methods are obtained by varying the level/threshold λ.

For each class, a single kernel bandwidth parameter was employed, and optimized by maximizing a cross-validation estimate of the AUC. Note that this ROC is different from the one used to evaluate the methods (see above). In particular, it still views P_0 as the null distribution, but now the alternative distribution is taken to be the uniform distribution P_2 for the inductive detector (see Section 5; effectively we use a uniform random sample of size n in place of the unlabeled data), P_X for SSND, and the appropriate $\tilde{P}_X$ for the hybrid methods (see Section 5). Thus, the test label information was not used at any stage (prior to validation) by any of the methods.

We also compared the learning methods for several values of π. For semi-supervised learning, we examined $\pi = 0.5, \pi = 0.2, \pi = 0.1$, and $\pi = 0.0$. For transductive learning, we examined $\pi = 0.5, \pi = 0.2$, and $\pi = 0.1$. The case $\pi = 0.0$ cannot be evaluated in the transductive paradigm because there are no positive examples in the unlabeled data. For each value of π, we discarded

just enough examples (either negative or positive) so that the desired proportion was achieved in the contaminated data. Note that the number of positive examples (novelties) in the contaminated sample could be very small. For the smallest data sets, in the semi-supervised setting and when $\pi = 0.1$, this number is less than 10.

We remark that the AUC is only one possible performance measure to assess our algorithms. An alternative choice, and one that is more directly connected to our theory, would be to select several different values of α, and compare the false negative and false positive rates, for each value of α, across all different data sets and algorithms. It would be straightforward in our experimental setup to calibrate the thresholds, using cross-validation for example, to achieve the desired false positive rate. We have adopted the AUC as a more concise alternative that in a sense averages the performance for Neyman-Pearson classification across the complete range of α.

8.2 Statistical Summaries and Methodology

The complete results are summarized in Tables 2 through 5. Tables 2 and 3 show the average AUC for each data set and experimental setting, for the semi-supervised and transductive paradigms respectively. The inductive method is labeled Ind. Our learning reduction is labeled SSND or TND depending on the setting. The hybrid methods are labeled H(p_n) in Tables 2-3, and Hybrid(p_n) in Tables 4-5.

We followed the methodology of Demšar (2006) for comparing algorithms across multiple data sets. For each data set and each experimental setting, the algorithms were ranked 1 (best) through 5 (worst) based on AUC. The Friedman test was used to determine, for each experimental setting, whether there was a significant difference in the average ranks of the five algorithms across the data sets. The average ranks and p-values are reported in Tables 4 and 5. The results indicate that there is a significant difference among the algorithms at the 0.1 significance level for all settings, with the exception of the transductive setting when $\pi = 0.1$.

When the Friedman test resulted in significant differences, we then performed a post-hoc Nemenyi test to assess when there was a significant difference between individual algorithms. For a five algorithm experiment on 18 data sets, with a significance level of 0.1, the critical difference for the Nemenyi test is 1.30. That is, when the average ranks of two algorithms differ by more than 1.30, their performance is deemed to be significantly different.

8.3 Analysis of Results

From the results presented in Tables 2-5, we draw the following conclusions.

1. The average ranks in Tables 4-5 conform to our expectations in many respects. SSND/TND outrank the inductive approach when $\pi = 0.5$, and inductive outranks SSND when $\pi = 0.0$. At the intermediate values $\pi = 0.1$ and 0.2, hybrid methods achieve the best ranking.

2. The average ranks also reveal that the performance of the hybrid methods vary according to the value of π. As π increases, the best performing hybrid has a correspondingly smaller amount of auxiliary uniform data appended to the unlabeled sample. This also conforms to our expectations.

data set	$\pi = 0.5$					$\pi = 0.2$				
	Ind.	SSND	H(1.0)	H(0.5)	H(0.1)	Ind.	SSND	H(1.0)	H(0.5)	H(0.1)
banana	0.924	0.939	0.931	0.933	0.936	0.924	0.915	0.924	0.923	0.921
breast-cancer	0.654	0.643	0.675	0.669	0.667	0.654	0.557	0.657	0.648	0.621
diabetes	0.744	0.782	0.770	0.772	0.776	0.744	0.684	0.724	0.727	0.717
flare-solar	0.674	0.661	0.664	0.660	0.662	0.674	0.629	0.641	0.643	0.642
german	0.628	0.703	0.693	0.696	0.704	0.628	0.582	0.633	0.632	0.636
heart	0.793	0.854	0.845	0.853	0.851	0.793	0.690	0.805	0.789	0.745
ringnorm	0.999	0.997	0.996	0.996	0.996	0.999	0.992	0.990	0.991	0.983
thyroid	0.985	0.966	0.964	0.967	0.955	0.985	0.889	0.929	0.940	0.943
titanic	0.628	0.643	0.636	0.644	0.643	0.628	0.612	0.636	0.634	0.628
twonorm	0.915	0.993	0.989	0.989	0.990	0.915	0.940	0.961	0.958	0.953
waveform	0.761	0.958	0.952	0.945	0.956	0.761	0.839	0.848	0.896	0.901
image	0.818	0.939	0.929	0.935	0.939	0.818	0.892	0.874	0.879	0.875
splice	0.415	0.935	0.905	0.921	0.932	0.415	0.702	0.613	0.764	0.785
ionosphere	0.256	0.926	0.839	0.921	0.922	0.256	0.695	0.475	0.607	0.704
mushrooms	0.945	1.000	1.000	1.000	1.000	0.945	0.999	0.999	0.999	0.999
sonar	0.688	0.752	0.757	0.764	0.764	0.688	0.595	0.682	0.683	0.646
adult	0.605	0.872	0.872	0.864	0.835	0.605	0.705	0.720	0.829	0.720
web	0.462	0.778	0.749	0.697	0.788	0.462	0.616	0.631	0.585	0.674

data set	$\pi = 0.1$					$\pi = 0.0$				
	Ind.	SSND	H(1.0)	H(0.5)	H(0.1)	Ind.	SSND	H(1.0)	H(0.5)	H(0.1)
banana	0.924	0.891	0.922	0.919	0.913	0.924	0.540	0.919	0.905	0.785
breast-cancer	0.654	0.515	0.643	0.633	0.575	0.654	0.556	0.640	0.628	0.568
diabetes	0.744	0.605	0.699	0.700	0.692	0.744	0.494	0.689	0.669	0.657
flare-solar	0.674	0.571	0.624	0.629	0.626	0.674	0.471	0.613	0.603	0.611
german	0.628	0.548	0.623	0.624	0.602	0.628	0.522	0.595	0.608	0.592
heart	0.793	0.593	0.778	0.776	0.688	0.793	0.506	0.759	0.750	0.620
ringnorm	0.999	0.984	0.981	0.986	0.991	0.999	0.478	0.958	0.978	0.985
thyroid	0.985	0.786	0.884	0.906	0.895	0.985	0.590	0.852	0.869	0.795
titanic	0.628	0.591	0.632	0.634	0.621	0.628	0.443	0.630	0.628	0.572
twonorm	0.915	0.931	0.945	0.934	0.923	0.915	0.480	0.894	0.879	0.860
waveform	0.761	0.801	0.815	0.822	0.806	0.761	0.487	0.736	0.727	0.705
image	0.818	0.769	0.824	0.836	0.851	0.818	0.431	0.634	0.696	0.780
splice	0.415	0.630	0.518	0.584	0.625	0.415	0.523	0.447	0.493	0.493
ionosphere	0.256	0.618	0.438	0.488	0.575	0.256	0.520	0.392	0.431	0.486
mushrooms	0.945	0.995	0.992	0.998	0.996	0.945	0.566	0.972	0.980	0.982
sonar	0.688	0.556	0.658	0.652	0.615	0.688	0.510	0.628	0.643	0.587
adult	0.605	0.627	0.659	0.666	0.626	0.605	0.505	0.558	0.556	0.572
web	0.462	0.554	0.584	0.544	0.611	0.462	0.557	0.553	0.523	0.564

Table 2: AUC values for five novelty detection algorithms in the semi-supervised setting. 'H' indicates a hybrid method.

data set	$\pi = 0.5$					$\pi = 0.2$				
	Ind.	TND	H(1.0)	H(0.5)	H(0.1)	Ind.	TND	H(1.0)	H(0.5)	H(0.1)
banana	0.924	0.938	0.931	0.932	0.935	0.924	0.915	0.923	0.923	0.919
breast-cancer	0.663	0.673	0.662	0.662	0.670	0.663	0.615	0.649	0.659	0.630
diabetes	0.742	0.784	0.776	0.779	0.788	0.742	0.708	0.728	0.725	0.727
flare-solar	0.673	0.686	0.683	0.684	0.684	0.673	0.661	0.658	0.662	0.666
german	0.633	0.739	0.709	0.711	0.714	0.633	0.617	0.632	0.637	0.636
heart	0.796	0.869	0.856	0.856	0.864	0.796	0.716	0.811	0.794	0.788
ringnorm	0.999	0.997	0.996	0.996	0.996	0.999	0.993	0.989	0.991	0.983
thyroid	0.984	0.976	0.978	0.979	0.974	0.984	0.957	0.962	0.955	0.962
titanic	0.629	0.667	0.646	0.658	0.661	0.629	0.642	0.641	0.658	0.645
twonorm	0.915	0.993	0.990	0.990	0.990	0.915	0.940	0.961	0.961	0.956
waveform	0.771	0.960	0.953	0.947	0.957	0.771	0.847	0.850	0.900	0.905
image	0.845	0.955	0.949	0.949	0.953	0.845	0.897	0.889	0.891	0.901
splice	0.416	0.941	0.913	0.930	0.939	0.416	0.716	0.623	0.769	0.820
ionosphere	0.254	0.953	0.844	0.931	0.952	0.254	0.714	0.413	0.633	0.746
mushrooms	0.945	1.000	1.000	1.000	1.000	0.945	0.999	0.999	0.999	0.999
sonar	0.683	0.757	0.767	0.778	0.781	0.683	0.615	0.678	0.683	0.662
adult	0.606	0.875	0.873	0.865	0.835	0.606	0.687	0.736	0.847	0.739
web	0.464	0.810	0.758	0.727	0.788	0.464	0.644	0.639	0.590	0.667

data set	$\pi = 0.1$				
	Ind.	TND	H(1.0)	H(0.5)	H(0.1)
banana	0.924	0.896	0.921	0.920	0.910
breast-cancer	0.663	0.564	0.687	0.642	0.598
diabetes	0.742	0.658	0.720	0.709	0.693
flare-solar	0.673	0.615	0.655	0.643	0.659
german	0.633	0.556	0.615	0.616	0.615
heart	0.796	0.626	0.792	0.784	0.729
ringnorm	0.999	0.985	0.973	0.986	0.992
thyroid	0.984	0.910	0.970	0.955	0.932
titanic	0.629	0.603	0.643	0.642	0.626
twonorm	0.915	0.933	0.943	0.937	0.923
waveform	0.771	0.813	0.821	0.823	0.808
image	0.845	0.888	0.870	0.871	0.880
splice	0.416	0.630	0.554	0.553	0.640
ionosphere	0.254	0.589	0.349	0.443	0.552
mushrooms	0.945	0.996	0.994	0.997	0.997
sonar	0.683	0.514	0.646	0.655	0.592
adult	0.606	0.658	0.681	0.684	0.629
web	0.464	0.567	0.573	0.538	0.604

Table 3: AUC values for five novelty detection algorithms in the transductive setting. 'H' indicates a hybrid method.

π	Inductive	SSND	Hybrid(1.0)	Hybrid(0.5)	Hybrid(0.1)	p-value
0.0	1.89	4.39	2.72	2.89	3.11	0.000
0.1	2.83	4.00	2.83	2.28	3.06	0.023
0.2	3.28	3.83	2.61	2.56	2.72	0.071
0.5	4.28	1.94	3.44	3.00	2.33	0.000

Table 4: The comparison of average ranks of the five algorithms in the semi-supervised setting, by the Friedman test. The critical difference of the post-hoc Nemenyi test is 1.30 at a significance level of 0.1.

π	Inductive	TND	Hybrid(1.0)	Hybrid(0.5)	Hybrid(0.1)	p-value
0.1	2.94	3.78	2.56	2.67	3.06	0.157
0.2	3.17	3.78	3.06	2.50	2.50	0.085
0.5	4.44	1.44	3.56	3.17	2.39	0.000

Table 5: The comparison of average ranks of the five algorithms in the transductive setting, by the Friedman test. The critical difference of the post-hoc Nemenyi test is 1.30 at a significance level of 0.1.

3. All tables indicate that the proposed methodology performs better in the transductive setting than the semi-supervised setting. A likely reason is that, in our experimental setup, TND sees twice as much unlabeled data as SSND.

4. When $\pi = 0.0$ in the semi-supervised experiments, SSND typically has an AUC around 0.5, which corresponds to random guessing. This makes sense, because it is essentially trying to classify between two realizations of the nominal distribution. From Tables 2 and 4 we see that the hybrid methods clearly improve upon SSND when $\pi = 0.0$.

5. For some data sets (splice, ionosphere, web), the inductive method does worse than random guessing, but our methods do not. In each case, our methods yield dramatic increases in AUC.

6. The benefits of unlabeled data increase with dimension. In particular, SSND and TND tend to perform much better relative to the inductive approach on data sets of dimension at least 18. This is especially evident in the second half of the data sets, which even show significant gains for $\pi = 0.1$. This trend suggests that as dimension increases, the assumption implicit in the inductive approach (that novelties are uniform where they overlap the support of the nominal distribution) breaks down.

Figure 1 depicts a sampling of results comparing the inductive and semi-supervised methods, and highlights the impact of dimension. The top graph shows ROCs for a two-dimensional data set where the two classes are fairly well separated, meaning the novelties lie in the tails of the nominal class, and $\pi = 0.5$. Not surprisingly, the inductive method is close to the semi-supervised method. The middle graph represents the 60-dimensional splice data set, where the inductive method does worse than random guessing, yet SSND does quite well. The bottom graph in Figure 1 shows the results for the 21-dimensional waveform data for $\pi = 0.1$. Here the assumptions of the inductive approach are also evidently violated to some degree.

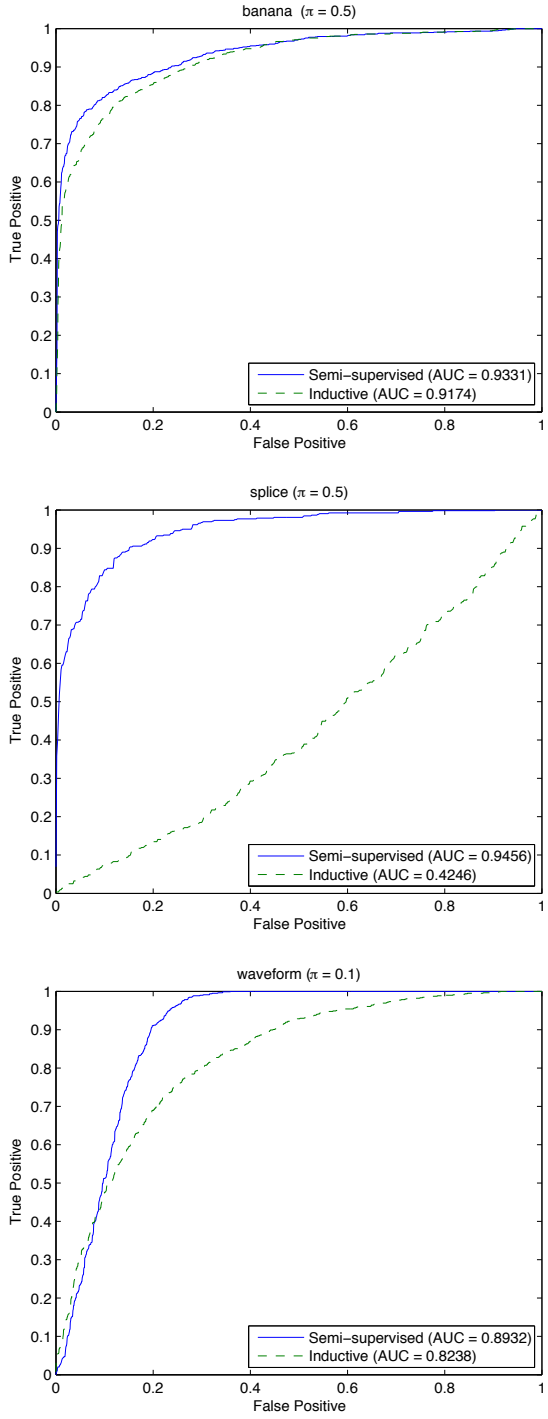


Figure 1: Illustrative results from the semi-supervised setting. Top: In the 2-dimensional banana data, the two classes are well separated, and the inductive approach fares well. Middle: In the 60-dimensional splice data, the inductive approach does worse than random guessing. Bottom: In the 21-dimensional waveform data, unlabeled data still offer gains when π is small (here 0.1).

9. Conclusions

We have shown that semi-supervised novelty detection reduces to Neyman-Pearson classification. This allows us to leverage known performance guarantees for NP classification algorithms, and to import practical algorithms. We have applied techniques from statistical learning theory, such as uniform deviation inequalities, to establish distribution free performance guarantees for SSND, as well as a distribution free lower bound and universally consistent estimator for π, and test for $\pi = 0$. Our approach optimally adapts to the unknown novelty distribution, unlike inductive approaches, which operate as if novelties are uniformly distributed. We also introduced a hybrid method that has the properties of SSND when $\pi > 0$, and effectively reverts to the inductive method when $\pi = 0$.

Our analysis strongly suggests that in novelty detection, unlike traditional binary classification, unlabeled data are essential for attaining optimal performance in terms of tight bounds, consistency, and rates of convergence. In an extensive experimental study, we found that the advantages of our approach are most pronounced for high dimensional data. Our analysis and experiments confirm some challenges that seem to be intrinsic to the SSND problem. In particular, SSND is more difficult for smaller π. Furthermore, estimating the novelty proportion π can become arbitrarily difficult as the nominal and novel distributions become increasingly similar.

Our methodology also provides general solutions to two well-studied problems in hypothesis testing. First, our lower bound on π translates immediately to a test for $\pi = 0$, which amounts to a distribution-free solution to the two-sample problem. Second, we also show that SSND provides a powerful generalization of standard multiple testing. Important problems for future work will include developing practical methodologies for these problems based on our theoretical framework.

Acknowledgments

C. Scott and G. Lee were supported in part by NSF Award No. 0830490. G. Blanchard was a researcher at the Weierstrass Institut (WIAS), Berlin, while taking part in this work, and acknowledges support of the PASCAL2 network of excellence of the European community, FP7 ICT-216886.

Appendix A. Proofs

The remaining proofs are now presented.

A.1 Proof of Theorem 2

For the first two claims of the theorem, we directly apply Theorem 3 of Scott and Nowak (2005) to the problem of NP classification of P_0 versus P_X , and obtain that for a suitable choice of constants c, c' we have with probability at least $1-\delta$:

$$R_0(\widehat{f_\tau}) - \alpha \leq c'\varepsilon_n \; ; \; R_X(\widehat{f_\tau}) - R_X(f^*) \leq c'\varepsilon_m \, .$$

From this, we deduce (3)-(4) by application of Theorem 1.

For the second claim, by application of Bayes' rule we have for any classifier f :

$$Q_0(f) = \frac{(1-\pi)(1-R_0(f))}{P_X(f(X)=0)} = \frac{(1-\pi)(1-R_0(f))}{\pi R_1(f) + (1-\pi)(1-R_0(f))}$$

and

$$Q_1(f) = \frac{\pi(1-R_1(f))}{P_X(f(X)=1)} = \frac{\pi(1-R_1(f))}{(1-\pi)R_0(f)+\pi(1-R_1(f))}\,.$$

Observe that for $a,b>0$ the function $\zeta(x,y) = \frac{a(1-x)}{by+a(1-x)}$ is decreasing in $y \in \mathbb{R}_+$ for $x \in (-\infty,1]$, and decreasing in $x \in [0,1+by/a)$ for $y \in \mathbb{R}_+$. Hence, using (3)-(4) and the fact that $R_i(f) \in [0,1]$, we derive a lower bound on $Q_0(\widehat{f_\tau})$ as follows:

$$\begin{aligned}
Q_0(\widehat{f_\tau}) &= \frac{(1-\pi)(1-R_0(\widehat{f_\tau}))}{\pi R_1(\widehat{f_\tau})+(1-\pi)(1-R_0(\widehat{f_\tau}))} \\
&\geq \frac{(1-\pi)(1-\alpha-c'\varepsilon_n)}{\pi(R_1(f^*)+c'\pi^{-1}(\varepsilon_n+\varepsilon_m))+(1-\pi)(1-R_0(f^*)-c'\varepsilon_n)} \\
&= \frac{(1-\pi)(1-\alpha-c'\varepsilon_n)}{P_X(f^*(X)=0)+c'(\varepsilon_m+\pi\varepsilon_n)} \\
&\geq \frac{(1-\pi)(1-\alpha)}{P_X(f^*(X)=0)+c'(\varepsilon_m+\pi\varepsilon_n)} - \frac{c'(1-\pi)\varepsilon_n}{P_X(f^*(X)=0)} \\
&\geq \frac{(1-\pi)(1-\alpha)-c'(1-\pi)\varepsilon_n}{P_X(f^*(X)=0)} - \frac{(1-\pi)(1-\alpha)c'(\varepsilon_m+\pi\varepsilon_n)}{P_X(f^*(X)=0)^2} \\
&\geq Q_0(f^*) - \frac{c'(\varepsilon_n+\varepsilon_m)}{P_X(f^*(X)=0)}\,.
\end{aligned}$$

The first inequality comes from the monotonicity properties of $\zeta(x,y)$ (applied first with respect to y, then x). The second is elementary. In the third inequality we used the fact that the function $g:\delta \mapsto g(\delta) = \frac{A}{B+\delta}$ is convex for A,B,δ positive and has derivative $-A/B^2$ in zero, so that $g(\delta) \geq \frac{A}{B} - \delta\frac{A}{B^2}$, with $A=(1-\pi)(1-\alpha), B = P_X(f^*(X)=0), \delta = c'(\varepsilon_m+\pi\varepsilon_n)$. In the last inequality we used (with the same definition for A,B) that $\frac{A}{B} = Q_0(f^*) \leq 1$.

The treatment for Q_1 is similar. Suppose first that

$$c'(\varepsilon_n+\varepsilon_m) < P_X(f^*(X)=1). \tag{15}$$

We then have

$$\begin{aligned}
Q_1(\widehat{f_\tau}) &= \frac{\pi(1-R_1(\widehat{f_\tau}))}{(1-\pi)R_0(\widehat{f_\tau})+\pi(1-R_1(\widehat{f_\tau}))} \\
&\geq \frac{\pi(1-R_1(f^*)-c'\pi^{-1}(\varepsilon_n+\varepsilon_m))}{(1-\pi)(\alpha+c'\varepsilon_n)+\pi(1-R_1(f^*)-c'\pi^{-1}(\varepsilon_n+\varepsilon_m))} \\
&= \frac{\pi(1-R_1(f^*))-c'(\varepsilon_n+\varepsilon_m)}{P_X(f^*(X)=1)-c'(\pi\varepsilon_n+\varepsilon_m)} \\
&\geq Q_1(f^*) - \frac{c'(\varepsilon_n+\varepsilon_m)}{P_X(f^*(X)=1)-c'(\varepsilon_n+\varepsilon_m)},
\end{aligned}$$

where we used again the monotonicity properties of $\zeta(x,y)$. Note that assumption (15) ensures that all denominators in the above chain of inequalities are positive, which is required for these

inequalities to hold. Now, since $Q_1(\widehat{f_\tau}) \geq 0$ and $Q_1(f^*) \leq 1$, the above implies

$$\begin{aligned} Q_1(\widehat{f_\tau}) &\geq Q_1(f^*) - \min\left(1, \frac{c'(\varepsilon_n + \varepsilon_m)}{P_X(f^*(X)=1) - c'(\varepsilon_n+\varepsilon_m)}\right) \\ &\geq Q_1(f^*) - \frac{2c'(\varepsilon_n+\varepsilon_m)}{P_X(f^*(X)=1)}; \end{aligned}$$

in the last inequality we used the fact that $\min(1, x/(1-x)) \leq 2x$ for $x \in [0,1)$ (with $x = c'(\varepsilon_n + \varepsilon_m)/P_X(f^*(X)=1)$). If (15) is not satisfied, then the last display still trivially holds since its RHS is nonpositive. This establishes (5) for $i = 1$.

A.2 Proof of Proposition 5

Let π^* be defined as

$$\pi^* := \inf\{\alpha \in [0,1] : \exists Q \text{ probability distribution: } P_X = (1-\alpha)P_0 + \alpha Q\}.$$

We want to establish that π^* satisfies the claims of the theorem (and in particular that the above infimum is a minimum). In the case $P_0 = P_X$, obviously we have $\pi^* = 0$ and this is a minimum. We now assume for the remainder of the proof that $P_0 \neq P_X$. Consider the Lebesgue decomposition $P_X = P_X^0 + P_X^\perp$ with $P_X^0 \ll P_0$ (that is, P_X^0 is absolutely continuous with respect to P_0) and $P_X^\perp \perp P_X^0$ (that is, $P_X^\perp$ and P_0 are mutually singular). Let $f = dP_X^0/dP_0$ and a be the essential infimum of f wrt. P_0. We claim that $\pi^* = 1 - a$. Observe first that

$$a \leq E_{X\sim P_0}[f(X)] = P_X^0(\mathcal{X}) \leq P_X(\mathcal{X}) = 1.$$

In particular, $a = 1$ must imply that the above inequalities are equalities, hence that $E_{X\sim P_0}[f] = a$. The latter can only be valid if $f = a = 1$ P_0-a.s., implying that $P_0 = P_X^0$, and further $P_0 = P_X$, which we excluded before. Therefore it holds that $a < 1$. Certainly we then have the valid decomposition

$$P_X = aP_0 + (1-a)P_1, \qquad P_1 := \left((1-a)^{-1}\left((f-a)P_0 + P_X^\perp\right)\right),$$

so that $\pi^* \leq 1 - a$.

By definition of singular measures there exists a measurable set D such that $P_0(D) = 1$ and $P_X^\perp(D) = 0$. Fix $\varepsilon > 0$; by definition of the essential infimum there exists a measurable set C such that $P_0(C) > 0$ and $f \leq a + \varepsilon$ P_0-a.s. on C. Put $A = C \cap D$. Then $P_0(A) = P_0(C) > 0$. Furthermore

$$\frac{P_1(A)}{P_0(A)} = \frac{E_{X\sim P_0}\left[(1-a)^{-1}(f-a)\mathbf{1}_{\{X\in A\}}\right]}{P_0(A)} \leq \varepsilon/(1-a).$$

Existence of a decomposition of the form $P_1 = (1-\gamma)Q + \gamma P_0$ implies that for any measurable set A, $P_1(A) \geq \gamma P_0(A)$. Hence the above implies that $\gamma = 0$ for any such decomposition, that is, P_1 must be a proper novelty distribution wrt. P_0. It also implies that for any $\varepsilon > 0$ there exists a measurable set A with $P_0(A) > 0$ and $P_X(A)/P_0(A) \leq a + \varepsilon$. Hence for any decomposition $P_X = (1-\alpha)P_0 + \alpha Q$, it must hold that $(1-\alpha) \leq a$, so that $\pi^* \geq 1 - a$. We thus established $\pi^* = 1 - a$ and the existence of the decomposition. Concerning the unicity, the decomposition established above implies that for any $\alpha \geq \pi^*$, $P_X = (1-\alpha)P_0 + \alpha Q$ holds with $Q = (1 - \frac{\pi^*}{\alpha})P_0 + \frac{\pi^*}{\alpha}P_1$. Note that for any fixed α, existence of a decomposition $P_X = (1-\alpha)P_0 + \alpha Q$ uniquely determines Q. Hence for $\alpha > \pi^*$ the corresponding Q is not a proper novelty distribution, and the only valid decomposition of P_X into P_0 and a proper novelty distribution is the one established previously.

A.3 Lemma Used in Proof of Theorem 6

For the proof of Theorem 6 we made use of the following auxiliary result:

Lemma 13 *Assume P_1 is a proper novelty distribution wrt. P_0. Then for any $\varepsilon > 0$ there exists a (deterministic) classifier f such that $R_0(f) < 1$ and*

$$\frac{R_1(f)}{1-R_0(f)} \leq \varepsilon\,.$$

Proof . Since P_1 is a proper novelty distribution wrt. P_0, reiterating the reasoning in the proof of Proposition 5 shows that there exists a measurable set A with $P_0(A) > 0$ and $P_1(A)/P_0(A) \leq \varepsilon$. Put $\alpha = 1 - P_0(A) < 1$. Consider the classifier $f(x) = \mathbf{1}_{\{x \in A^c\}}$. Then $R_0(f) = P_0(f=1) = \alpha$, while

$$0 \leq R_1(f) = P_1(f=0) = P_1(A) \leq \varepsilon(1-\alpha)\,. \tag{16}$$

This leads to the desired conclusion.

■

A.4 Proof of Theorem 8

By application of Lemma 13, for any $\varepsilon > 0$ there exists a classifier f^* such that $\frac{R_1(f^*)}{1-R_0(f^*)} \leq \varepsilon$. Then we have as in the proof of Theorem 6:

$$1 - \frac{R_X(f^*)}{1-R_0(f^*)} = \pi\left(1 - \frac{R_1(f^*)}{1-R_0(f^*)}\right) \geq \pi(1-\varepsilon)\,.$$

Fix $\gamma > 0$ and define $\widetilde{P} = \frac{1}{2}(P_0 + P_1)$. Using the assumption of universal approximation, pick k such that there exists $f_k^* \in \mathcal{F}_k$ with $\widetilde{P}(f_k^*(X) \neq f^*(X)) \leq \gamma$. Since $\widetilde{P} \geq \frac{1}{2}P_0$ and $\widetilde{P} \geq \frac{1}{2}P_1$ this implies also $P_0(f_k^*(X) \neq f^*(X)) \leq 2\gamma$ as well as $P_X(f_k^*(X) \neq f^*(X)) \leq 2\gamma$.

From now we only work in the class $\mathcal{F}_k$ and so we omit the parameters in the notation $\varepsilon_i \equiv \varepsilon_i(\mathcal{F}_k, \delta k^{-2})$. By the union bound, the uniform control of the form (11) is valid simultaneously for all $\mathcal{F}_k$, with probability $1 - c\delta$ (with $c = \pi^2/6$). Hence with probability $1 - c\delta = 1 - c(mn)^{-2}$, we have

$$\widehat{R}_0(f_k^*) \leq R_0(f_k^*) + \varepsilon_m \leq R_0(f^*) + 2\gamma + \varepsilon_m\,,$$

and also

$$\widehat{R}_X(f_k^*) \leq R_X(f_k^*) + \varepsilon_n \leq R_X(f^*) + 2\gamma + \varepsilon_n\,.$$

From this we deduce that with probability $1 - c(mn)^{-2}$:

$$\widehat{\pi}^-(\delta) \geq \widehat{\pi}^-(\mathcal{F}_k, (mn)^{-2}k^{-2}) \geq 1 - \frac{R_X(f^*) + 2\gamma + 2\varepsilon_n}{1 - R_0(f^*) - 2\gamma - 2\varepsilon_m}\,.$$

Since $\varepsilon_n, \varepsilon_m$ go to zero as $\min(m,n)$ goes to infinity we deduce that a.s. (using the Borel-Cantelli lemma, and the fact that the error probabilities are summable over $(m,n) \in \mathbb{N}^2$)

$$\liminf_{\min(m,n)\to\infty} \widehat{\pi}^-(\delta) \geq 1 - \frac{R_X(f^*) + 2\gamma}{1 - R_0(f^*) - 2\gamma} \geq \pi(1-\varepsilon)\frac{1 - R_0(f^*)}{1 - R_0(f^*) - 2\gamma} - \frac{4\gamma}{1 - R_0(f^*) - 2\gamma}\,.$$

Taking the limit of the above as $\gamma \to 0$ (for fixed ε and f^*), then as $\varepsilon \to 0$, leads to the conclusion.

A.5 Proof of Theorem 9

The principle of the argument is relatively standard and can be sketched as follows. Assume there exists a non-trivial upper confidence bound $\widehat{\pi}^+(\delta)$ on the proportion of anomalies. From the non-triviality assumption, there exists a generating distribution P and a set of samples $\mathcal{B}$ such that $\widehat{\pi}^+(\delta) < 1$ for all samples in $\mathcal{B}$ while $P(\mathcal{B}) > \delta$. We will construct below an alternate generating distribution $\widetilde{P}$ and set of samples $\widetilde{\mathcal{B}} \subset \mathcal{B}$ which are very close to P and $\mathcal{B}$, in particular satisfying $\widetilde{P}(\widetilde{\mathcal{B}}) > \delta$; however the proportion of anomalies for $\widetilde{P}$ is 1, contradicting the universality of $\widehat{\pi}^+$ since $\widehat{\pi}^+(\delta) < 1$ for all samples in $\widetilde{\mathcal{B}}$.

Let P_0, P_1, δ, π be given by the non-triviality assumption and $P := P_0^{\otimes m} \otimes P_X^{\otimes n}$ denote correspondingly the joint distribution of nominal and contaminated data. Fix some $\gamma > 0$ and a set D such that $0 < P_0(D) < \gamma$ (such a set always exists by the assumption that P_0 is weakly diffuse). Put $A = D^c$, so that $1 - \gamma < P_0(A) < 1$. Consider the distribution P_0 conditional to belonging to A, given by $\frac{\mathbf{1}_{x \in A}}{P_0(A)} P_0$. Since it has its support strictly included in the support of P_0, it is a proper novelty distribution with respect to P_0. Therefore, since P_1 is also a proper novelty distribution with respect to P_0, so is $\widetilde{P}_1 := (1-\pi)\frac{\mathbf{1}_{x \in A}}{P_0(A)} P_0 + \pi P_1$.

Now consider the novelty detection problem with nominal distribution P_0, novelty distribution $\widetilde{P}_1$, and novelty proportion $\widetilde{\pi} = 1$, so that $\widetilde{P}_X = (1-\widetilde{\pi})P_0 + \widetilde{\pi}\widetilde{P}_1 = \widetilde{P}_1$. Finally, define the modified joint distribution on nominal and contaminated data $\widetilde{P} = P_0^{\otimes m} \otimes \widetilde{P}_X^{\otimes n}$.

By the non-triviality assumption, there exists a set $\mathcal{B}$ of (m,n) samples such that $\widehat{\pi}^+(\delta) < 1$ on the set $\mathcal{B}$ and $P(\mathcal{B}) = \delta_0 > \delta$. Denote $\mathcal{A} = \mathcal{X}^m \times A^n$. By assumption, $P(\mathcal{A}) \geq (1-\gamma)^n$; furthermore from the construction of $\widetilde{P}$ it can be verified straightforwardly that for any set $\mathcal{C} \subset \mathcal{A}$, $\widetilde{P}(\mathcal{C}) \geq P(\mathcal{C})$. Define now $\widetilde{\mathcal{B}} = \mathcal{B} \cap \mathcal{A}$; we have

$$\widetilde{P}(\widetilde{\mathcal{B}}) \geq P(\widetilde{\mathcal{B}}) \geq \delta_0 - (1 - (1-\gamma)^n).$$

Hence for γ small enough, we have $\widetilde{P}(\widetilde{\mathcal{B}}) > \delta$ which contradicts the fact that $\widehat{\pi}^+(\delta)$ is a $1-\delta$ confidence upper bound, since on $\widetilde{\mathcal{B}}$ we have $\widehat{\pi}^+(\delta) < 1 = \widetilde{\pi}$.

A.6 Proof of Corollary 10

Put

$$\gamma := \sup_{P \in \mathcal{P}(m,n)} E\left[\left|\widehat{\pi} - \pi\right|^p\right].$$

By Markov's inequality, it means that for any generating distribution P,

$$P\left(\pi > \widehat{\pi} + \left(\frac{\gamma}{\delta}\right)^{\frac{1}{p}}\right) \leq \delta.$$

Hence $\widehat{\pi}^+(\delta) = \widehat{\pi} + \left(\frac{\gamma}{\delta}\right)^{\frac{1}{p}}$ is a distribution-free upper confidence bound on π.

By theorem (9), this bound must be trivial. We investigate the consequences of this fact. Let us consider any generating distribution $P \in \mathcal{P}(m,n)$ for which $\pi = 0$, and the nominal distribution P_0 is weakly diffuse (it is possible to find such a P_0 since $\mathcal{X}$ is infinite). Assume $\delta < \frac{1}{2}$ and fix some $\delta_0 \in (\delta, 1-\delta)$. Markov's inequality again implies that for this distribution,

$$P\left(\widehat{\pi} \geq \left(\frac{\gamma}{\delta_0}\right)^{\frac{1}{p}}\right) \leq \delta_0,$$

so that

$$P\left(\widehat{\pi}^{+}(\delta) \geq 2\left(\frac{\gamma}{\delta}\right)^{\frac{1}{p}}\right) \leq \delta_0 \leq 1-\delta.$$

Choosing $\delta_0 = 1/2, \delta = 1/3$, the above implies that $\widehat{\pi}^{+}(\delta)$ would be non-trivial if $\gamma < 2^{-p}/3$. Thus we must conclude that $\gamma \geq c(p) := 2^{-p}/3$.

Appendix B. Randomized classifiers and ROCs

In this appendix we review some properties of ROCs that are relevant to our setting. These should be considered well-known (see for example the paper of Fawcett, 2006, and references therein, for an overview of ROC analysis over sets of classifiers). For the sake of completeness, we give here statements and short proofs that are tailored to correspond precisely to the context considered in the main body of the paper.

Let $\mathcal{F}$ be a fixed set of classifiers, and recall the Neyman-Pearson classification optimization problem (1), restated here for convenience:

$$\begin{aligned} R^*_{1,\alpha}(\mathcal{F}) := \inf_{f\in\mathcal{F}} \; & R_1(f) \\ \text{s.t. } & R_0(f) \leq \alpha\,. \end{aligned} \tag{1}$$

We call optimal ROC of P_1 versus P_0 for set $\mathcal{F}$ the function $\alpha \in [0,1] \mapsto 1 - R^*_{1,\alpha}(\mathcal{F}) \in [0,1]$.

For reference, first consider a very "regular" case where $\mathcal{F}$ is the set of all possible deterministic classifiers, and one assumes that both class probabilities P_0, P_1 have densities h_0, h_1 with respect to some reference measure, such that the likelihood ratio $F(x) = \frac{h_1(x)}{h_0(x)}$ is continuous with $\inf F = 0$ and $\sup F = +\infty$. Then the optimal solutions f^*_α of (1) are indicators of sets of the form

$$C_\lambda = \left\{ x \in \mathcal{X} : \frac{h_1(x)}{h_0(x)} \geq \lambda \right\},$$

with $\lambda(\alpha)$ such that $P_0(C_\lambda) = \alpha$. In this case $R^*_{1,\alpha}(\mathcal{F}) = P_1(C_{\lambda(\alpha)})$ and it can be shown that the ROC is continuous, nondecreasing and concave between the points $(0,0)$ and $(1,1)$. In particular in this case it holds that $R_0(f^*_\alpha) = \alpha$.

When some of the above assumptions are not satisfied, for example if we consider an arbitrary subset $\mathcal{F}$ of classifiers (which is of course necessary for adequate estimation error control), or the probability distributions P_0 and P_1 have atoms (which is the case in practice for empirical distributions), some of these properties may fail to hold. While it is clear that the optimal ROC is always a nondecreasing function, it might fail to be concave, and the optimal solution might have $R_0(f^*_\alpha) < \alpha$. This is for example obviously the case if $\mathcal{F}$ is a finite set of classifiers, in which case the ROC is a step function and $R_0(f)$ can only take finitely many values.

We are interested in the following regularity properties depending on $\mathcal{F}, P_0$ and P_1:

(A') For any $\alpha \in (0,1)$, there exists a sequence $f_n \in \mathcal{F}$ such that $R_0(f_n) = \alpha$ and $R_1(f_n) \to R^*_{1,\alpha}(\mathcal{F})$.

(B) The function $\alpha \mapsto R^*_{1,\alpha}(\mathcal{F})/(1-\alpha)$ is nondecreasing on $[0,1]$.

Note that for simplicity of exposition, in the main body of the paper we simplified property **(A')** into **(A)**, where the sequence f_n is replaced by its limit, assumed to belong to the considered set of classifiers. Our results still hold under **(A')** with straightforward modifications of the proofs.

Condition **(B)** states that the slope of the line joining the point of the optimal ROC at α and the point $(1,1)$ is nonincreasing in α; this assumption is weaker than concavity of the ROC. It is relevant for the discussion in the final paragraph below, related to our result on precision.

To ensure regularity properties of the ROC, a standard device is to extend the class $\mathcal{F}$ and consider *randomized* classifiers, whose output is not a deterministic function, but a Bernoulli variable with probability depending on the point x. Formally this amounts to allowing a classifier f to take real values in the interval $[0,1]$; now for a given x the final decision $D(f,x)$ is a Bernoulli variable (independent of everything else) taking value 1 with probability $f(x)$ and 0 with probability $1-f(x)$. In this setting the error probabilities become for $y=0,1$:

$$R_y(f) := P_y(D(f,X) \neq y) = E_y[|f(X)-y|],$$

where E_y is a shortcut for $E_{X\sim P_y}$. Although the function f itself remains strictly speaking deterministic, in view of the above interpretation we will call with some abuse f a deterministic classifier if it takes values in $\{0,1\}$ and randomized classifier if takes values in $[0,1]$.

We consider two types of extensions of a (usually deterministic) class $\mathcal{F}$, the first one is the convex hull of $\mathcal{F}$, or *full randomization*,

$$\overline{\mathcal{F}} = \left\{ g \middle| g = \sum_{i=1}^{N} \lambda_i f_i ; N \in \mathbb{N}, f_i \in \mathcal{F}, \lambda_i \geq 0 \text{ for } 1 \leq i \leq N, \sum_{i=1}^{N} \lambda_i = 1 \right\}.$$

The second is given by

$$\mathcal{F}^+ = \{ g | g = \lambda f + (1-\lambda), f \in \mathcal{F}, \lambda \in [0,1] \},$$

where the randomization is limited to convex interpolation between one classifier of the base class and the constant classifier equal to 1.

The following standard lemma summarizes the properties of the optimal ROC curve for these extended classes:

Lemma 14 *Let $\mathcal{F}$ be a set of deterministic classifiers containing the constant classifier equal to zero, and let P_0, P_1 be arbitrary distributions on $\mathcal{X}$. Then assumptions* **(A')** *and* **(B)** *are met when considering optimization problem* (1) *over either $\overline{\mathcal{F}}$ or $\mathcal{F}^+$. The optimal ROC for the set $\overline{\mathcal{F}}$ is concave.*

Proof . The fact that the constant zero classifier belongs to $\mathcal{F}$ ensures that the infimum in (1) (over either of the classes $\mathcal{F}$, $\mathcal{F}^+$ or $\overline{\mathcal{F}}$) is not taken over an empty set and exists. The definition of an infimum ensures that there exists a sequence g_n of elements of $\mathcal{F}^+$ such that $R_0(g_n) \leq \alpha$ and $R_1(g_n) \searrow R^*_{1,\alpha}(\mathcal{F}^+)$. Then putting $\lambda_n = (1-\alpha)/(1-R_0(g_n))$, the sequence $f_n = \lambda_n g_n + (1-\lambda_n)$ belongs to $\mathcal{F}^+$, and is such that

$$R_0(f_n) = E_0[f_n] = \frac{(1-\alpha)}{(1-R_0(g_n))} R_0(g_n) + \left(1 - \frac{(1-\alpha)}{(1-R_0(g_n))}\right) = \alpha$$

while

$$R^*_{1,\alpha}(\mathcal{F}^+) \leq R_1(f_n) = 1 - E_1[f_n] = \lambda_n R_1(g_n) \leq R_1(g_n) \searrow R^*_{1,\alpha}(\mathcal{F}^+),$$

and thus ensures **(A')**. The same reasoning applies to $\overline{\mathcal{F}}$.

For property **(B)**, consider a sequence (f_n) from property **(A')**, a number $\beta \in [\alpha, 1]$ and $h_n = (1-\zeta)f_n + \zeta$ where $\zeta = (1-\beta)/(1-\alpha) \in [0,1]$. Then $h_n \in \mathcal{F}^+$, $R_0(h_n) = \beta$ and $R_1(h_n) = (1-\zeta)R_1(f_n) + \zeta$. Letting n grow to infinity we obtain $R^*_{1,\beta}(\mathcal{F}^+) \leq (1-\zeta)R^*_{1,\alpha}(\mathcal{F}^+) + \zeta$ which in turn implies (**B**).

In the case of $\overline{\mathcal{F}}$, similarly consider sequences $f_{n,1}, f_{n,2}$ like above for $\alpha = \alpha_1$, resp. $\alpha = \alpha_2$ with $\alpha_2 \geq \alpha_1$; for any $\beta \in [\alpha_1, \alpha_2]$, write $\beta = \lambda\alpha_1 + (1-\lambda)\alpha_2$; correspondingly the sequence $\lambda f_{n,1} + (1-\lambda)f_{n,2}$ belongs to $\overline{\mathcal{F}}$ and ensures that $R^*_{1,\beta}(\overline{\mathcal{F}}) \leq \lambda R^*_{1,\alpha_1}(\overline{\mathcal{F}}) + (1-\lambda)R^*_{1,\alpha_2}(\overline{\mathcal{F}})$ that is, the optimal ROC for $\overline{\mathcal{F}}$ is concave. ■

Concerning estimation error control for the extended classes, a uniform control of the error on the base class is sufficient, since it carries over to the extended classes by convex combination. To be more specific, let us consider for example the estimation of $R_0(f)$ uniformly over $f \in \overline{\mathcal{F}}$. The empirical counterpart of $R_0(f)$ is given by

$$\widehat{R}_0(f) := \widehat{P}_0(D(f,X) \neq 0) = \widehat{E}_0[f(X)],$$

where $\widehat{E}_0$ denotes the empirical expectation on the nominal sample. By definition of $\overline{\mathcal{F}}$, f can be written $f = \sum_{i=1}^N \lambda_i f_i$ with $\sum_{i=1}^N \lambda_i = 1$ and $\lambda_i \geq 0, f_i \in \mathcal{F}$ for $1 \leq i \leq N$, and thus the estimation error is controlled as follows:

$$\begin{aligned}\left|R_0(f) - \widehat{R}_0(f)\right| = \left|E_0[f(X)] - \widehat{E}_0[f(X)]\right| &\leq \sum_{i=1}^N |\lambda_i| \left|P_0(f_i(X)=1) - \widehat{P}_0(f_i(X)=1)\right| \\ &\leq \sup_{f\in\mathcal{F}} \left|P_0(f(X)=1) - \widehat{P}_0(f(X)=1)\right|,\end{aligned}$$

Therefore, if an error control of the form (11) holds over the base class $\mathcal{F}$ (for example if it is a VC-class), then the same type of bound holds for quantities of interest over the extended classes $\mathcal{F}^+$ and $\overline{\mathcal{F}}$.

For practical purposes, it might be significantly more difficult to find the solution of the (empirical version of) (1) for randomized classes and in particular for the fully randomized extension $\overline{\mathcal{F}}$. An advantage of the more limited form of randomization is that optimization problem (1) over $\mathcal{F}^+$ can be rewritten equivalently as an optimization problem over the original class, namely as

$$\inf_{h\in\mathcal{F}} \frac{R_1(h)}{1-R_0(h)} \qquad \text{s.t. } R_0(h) \leq \alpha. \tag{17}$$

To see why, assume for simplicity of exposition that **(A)** rather than **(A')** is satisfied. Then the optimization problem (1) over $\mathcal{F}^+$ is attained for some randomized classifier f^*; by construction f^* is of the form $f^* = \lambda h^* + (1-\lambda)$ for some $\lambda \in [0,1]$ and $h^* \in \mathcal{F}$. By property **(A)** we can assume $R_0(f^*) = \alpha$, which entails $\lambda = (1-\alpha)/(1-R_0(h^*))$ and $R_1(f^*) = (1-\alpha)R_1(h^*)/(1-R_0(h^*))$, hence the equivalence with (17) (with the above relation between f^* and h^*).

Finally, in general we can interpret the optimization problem (17) as a maximization of the class 0 precision,

$$Q_0(f) = P_{XY}(Y=0|f(X)=0) = \frac{(1-\pi)(1-R_0(f))}{(1-\pi)(1-R_0(f)) + \pi R_1(f)} = \frac{(1-\pi)}{(1-\pi) + \pi\frac{R_1(f)}{1-R_0(f)}},$$

under the constraint $R_0(f) \leq \alpha$, since the above display shows that $Q_0(f)$ is a decreasing function of the ratio $R_1(f)/(1-R_0(f))$. In general if properties **(A)** and **(B)** are satisfied for the considered class, then it is easy to see that the solutions to (1) and (17) coincide, so that the same classifier f^* achieves the minimum FNR and class 0 precision under the constraint on the FPR.

References

S. Ben-David, T. Lu, and D. Pál. Does unlabeled data provably help? Worst-case analysis of the sample complexity of semi-supervised learning. In R. Servedio and T. Zhang, editors, *Proc. 21st Annual Conference on Learning Theory (COLT)*, pages 33–44, Helsinki, 2008.

A. Beygelzimer, V. Dani, T. Hayes, J. Langford, and B. Zadrozny. Error-limiting reductions between classification tasks. In L. De Raedt and S. Wrobel, editors, *Proceedings of the 22nd International Machine Learning Conference (ICML)*. ACM Press, 2005.

A. Cannon, J. Howse, D. Hush, and C. Scovel. Learning with the Neyman-Pearson and min-max criteria. Technical Report LA-UR 02-2951, Los Alamos National Laboratory, 2002.

C. C. Chang and C. J. Lin. LIBSVM : A library for support vector machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

Z. Chi. On the performance of FDR control: Constraints and a partial solution. *Ann. Stat.*, 35(4): 1409–1431, 2007.

Z. Chi. False discovery rate control with multivariate p-values. *Electronic Journal of Statistics*, 2: 368–411, 2008.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Machine Learning Research*, 7:1–30, 2006.

F. Denis. PAC learning from positive statistical queries. In *Proc. 9th Int. Conf. on Algorithmic Learning Theory (ALT)*, pages 112–126, Otzenhausen, Germany, 1998.

F. Denis, R. Gilleron, and F. Letouzey. Learning from positive and unlabeled examples. *Theoretical Computer Science*, 348(1):70–83, 2005.

L. Devroye. Any discrimination rule can have an arbitrarily bad probability of error for finite sample size. *IEEE Trans. Patt. Anal. Mach. Intell.*, 4:154–157, 1982.

B. Efron, R. Tibshirani, J.D. Storey, and V. Tusher. Empirical Bayes analysis of a microarray experiment. *Journal of the American Statistical Association*, 96:1151–1160, 2001.

R. El-Yaniv and M. Nisenson. Optimal single-class classification strategies. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Adv. in Neural Inform. Proc. Systems 19*. MIT Press, Cambridge, MA, 2007.

C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD08)*, pages 213–220, 2008.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

C. Genovese and L. Wasserman. A stochastic process approach to false discovery control. *Annals of Statistics*, 32(3):1035–1061, 2004.

D. Ghosh. Assessing significance of peptide spectrum matches in proteomics: A multiple testing approach. *Statistics in Biosciences*, 1:199–213, 2009.

D. Ghosh and A. M. Chinnaiyan. Genomic outlier profile analysis: Mixture models, null hypotheses, and nonparametric estimation. *Biostatistics*, 10:60–69, 2009.

A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample problem. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 513–520. MIT Press, Cambridge, MA, 2007.

A. Hero. Geometric entropy minimization for anomaly detection and localization. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Adv. in Neural Inform. Proc. Systems 19*. MIT Press, Cambridge, MA, 2007.

J. Lafferty and L. Wasserman. Statistical analysis of semi-supervised regression. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 801–808. MIT Press, Cambridge, MA, 2008.

W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *Proc. 20th Int. Conf. on Machine Learning (ICML)*, pages 448–455, Washington, DC, 2003.

E. Lehmann. *Testing Statistical Hypotheses*. Wiley, New York, 1986.

B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *Proc. 19th Int. Conf. Machine Learning (ICML)*, pages 387–394, Sydney, Australia, 2002.

B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. In *Proc. 3rd IEEE Int. Conf. on Data Mining (ICDM)*, pages 179–188, Melbourne, FL, 2003.

K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12:181–201, 2001.

P. Rigollet. Generalization error bounds in semi-supervised classification under the cluster assumption. *J. Machine Learning Research*, 8:1369–1392, 2007.

B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1472, 2001.

C. Scott and G. Blanchard. Novelty detection: Unlabeled data definitely help. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS) 2009*, pages 464–471, Clearwater Beach, Florida, 2009. JMLR: W&CP 5.

C. Scott and R. Nowak. A Neyman-Pearson approach to statistical learning. *IEEE Trans. Inform. Theory*, 51(11):3806–3819, 2005.

C. Scott and R. Nowak. Learning minimum volume sets. *J. Machine Learning Res.*, 7:665–704, 2006.

A. Singh, R. Nowak, and X. Zhu. Unlabeled data: Now it helps, now it doesn't. *Proc. Neural Information Processing Systems 21 –* NIPS '08, 2009.

A. Smola, L. Song, and C. H. Teo. Relative novelty detection. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS) 2009*, pages 536–543, Clearwater Beach, Florida, 2009. JMLR: W&CP 5.

D. Steinberg and N. S. Cardell. Estimating logistic regression models when the dependent variable has no variance. *Communications in Statistics - Theory and Methods*, 21:423–450, 1992.

I. Steinwart, D. Hush, and C. Scovel. A classification framework for anomaly detection. *J. Machine Learning Research*, 6:211–232, 2005.

J. D. Storey. The positive false discovery rate: A Bayesian interpretation and the q-value. *Annals of Statistics*, 31:6:2013–2035, 2003.

W. Sun and T. Cai. Oracle and adaptive compound decision rules for false discovery rate control. *J. Amer. Statist. Assoc.*, 102(479):901–912, 2007.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

R. Vert and J.-P. Vert. Consistency and convergence rates of one-class SVM and related algorithms. *J. Machine Learning Research*, pages 817–854, 2006.

G. Ward, T. Hastie, S. Barry, J. Elith, and J. R. Leathwick. Presence-only data and the EM algorithm. *Biometrics*, 65:554–564, 2009.

B. Zhang and W. Zuo. Learning from positive and unlabeled examples: A survey. In *Proceeding of the IEEE International Symposium on Information Processing (ISIP)*, pages 650–654, 2008.

D. Zhang and W. S. Lee. A simple probabilistic approach to learning from positive and unlabeled examples. In *Proc. 5th Annual UK Workshop on Comp. Intell. (UKCI)*, London, UK, 2005.

Journal of Machine Learning Research 11 (2010) 3011-3015 Submitted 8/10; Revised 9/10; Published 11/10

Gaussian Processes for Machine Learning (GPML) Toolbox

Carl Edward Rasmussen* CER54@CAM.AC.UK
Department of Engineering
University of Cambridge
Trumpington Street
Cambridge, CB2 1PZ, UK

Hannes Nickisch HN@TUE.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstraße 38
72076 Tübingen, Germany

Editor: Sören Sonnenburg

Abstract

The GPML toolbox provides a wide range of functionality for Gaussian process (GP) inference and prediction. GPs are specified by mean and covariance functions; we offer a library of simple mean and covariance functions and mechanisms to compose more complex ones. Several likelihood functions are supported including Gaussian and heavy-tailed for regression as well as others suitable for classification. Finally, a range of inference methods is provided, including exact and variational inference, Expectation Propagation, and Laplace's method dealing with non-Gaussian likelihoods and FITC for dealing with large regression tasks.

Keywords: Gaussian processes, nonparametric Bayes, probabilistic regression and classification

Gaussian processes (GPs) (Rasmussen and Williams, 2006) have convenient properties for many modelling tasks in machine learning and statistics. They can be used to specify distributions over functions without having to commit to a specific functional form. Applications range from regression over classification to reinforcement learning, spatial models, survival and other time series[1] models. Predictions of GP models come with a natural confidence measure: predictive error-bars.

Although the implementation of the basic principles in the simplest case is straight forward, various complicating features are often desired in practice. For example, a GP is determined by a *mean function* and a *covariance function*, but these functions are mostly difficult to specify fully a priori, and typically they are given in terms of *hyperparameters*, that is, parameters which have to be inferred. Another source of difficulty is the *likelihood function*. For Gaussian likelihoods, inference is analytically tractable; however, in many tasks, Gaussian likelihoods are not appropriate, and approximate inference methods such as Expectation Propagation (EP) (Minka, 2001), Laplace's approximation (LA) (Williams and Barber, 1998) and variational bounds (VB) (Gibbs and MacKay, 2000) become necessary (Nickisch and Rasmussen, 2008). In case of large training data, approximations (Candela and Rasmussen, 2005) like FITC (Snelson and Ghahramani, 2006) are needed.

The GPML toolbox is designed to overcome these hurdles with its variety of mean, covariance and likelihood functions as well as inference methods, while being simple to use and easy to extend.

∗. Also at Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany.

1. Note, that here we typically think of GPs with a more general index set than time.

1. Implementation

The GPML toolbox can be obtained from `http://gaussianprocess.org/gpml/code/matlab/` and also `http://mloss.org/software/view/263/` under the FreeBSD license. Based on simple interfaces for covariance, mean, likelihood functions as well as inference methods, we offer full compatibility to both Matlab 7.x[2] and GNU Octave 3.2.x.[3] Special attention has been given to properly disentangle covariance, likelihood and mean hyperparameters. Also, care has been taken to avoid numerical inaccuracies, for example, safe likelihood evaluations for extreme inputs and stable matrix operations. For example, the covariance matrix $\mathbf{K}$ can become numerically close to singular making its naive inversion numerically unsafe. We handle these situations in a principled way[4] such that Cholesky decompositions are computed of well-conditioned matrices only. As a result, our code shows a high level of robustness along the full spectrum of possible hyperparameters. The focus of the toolbox is on approximate inference using dense matrix algebra. We currently do not support covariance matrix approximation techniques to deal with large numbers of training examples n. Looking at the (growing) body of literature on sparse approximations, this knowledge is still somewhat in flux, and consensus on the best approaches has not yet been reached.

We provide stable and modular code checked by an exhaustive suite of test cases. A single function `gp.m` serves as main interface to the user—it can make inference and predictions and allows the mean, covariance and likelihood function as well as the inference methods to be specified freely.

Furthermore, `gp.m` enables convenient learning of the hyperparameters by maximising the log marginal likelihood $\ln Z$. One of the particularly appealing properties of GP models is that principled and practical approaches exist for learning the parameters of mean, covariance and likelihood functions. Good adaptation of such parameters can be essential to obtain both high quality predictions and insights into the properties of the data. The GPML toolbox is particularly flexible, including a large library of different covariance and mean functions, and flexible ways to combine these into more expressive, specialised functions. The user can choose between two gradient-based optimisers: one uses conjugate gradients (CG)[5] and the other one relies on a quasi-Newton scheme.[6] Computing the derivatives w.r.t. hyperparameters $\frac{\partial}{\partial \theta_i} \ln Z$ with `gp.m` does not need any extra programming effort; every inference method automatically collects the respective derivatives from the mean, covariance and likelihood functions and passes them to `gp.m`.

Our documentation comes in two pieces: a hypertext user documentation[7] `doc/index.html` with examples and code browsing and a technical documentation[8] `doc/manual.pdf` focusing on the interfaces and more technical issues. A casual user will use the hypertext document to quickly get his data analysed, however a power user will consult the pdf document once he wants to include his own mean, covariance, likelihood and inference routines or learn about implementation details.

2. Matlab is available from MathWorks, `http://www.mathworks.com/`.
3. Octave is available from the Free Software Foundation, `http://www.gnu.org/software/octave/`.
4. We do not consider the "blind" addition of a "small ridge" to $\mathbf{K}$ a principled way.
5. Carl Rasmussen's code is available at `http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/`.
6. Peter Carbonetto's wrapper can be found at `http://www.cs.ubc.ca/~pcarbo/lbfgsb-for-matlab.html`.
7. Documentation can be found at `http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html`.
8. Technical docs are available at `http://www.gaussianprocess.org/gpml/code/matlab/doc/manual.pdf`.

2. The GPML Toolbox

We illustrate the modular structure of the GPML toolbox by means of a simple code example. GPs are used to formalise and update knowledge about distributions over functions. A GP prior distribution on an unknown latent function $f \sim \mathcal{GP}(m_\phi(\mathbf{x}), k_\psi(\mathbf{x},\mathbf{x}'))$, consists of a mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$, and a covariance function $k(\mathbf{x},\mathbf{x}) = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$, both of which typically contain hyperparameters ϕ and ψ, which we want to fit in the light of data. We generally assume independent observations, that is, input/output pairs $(\mathbf{x}_i, y_i)$ of f with joint likelihood $\mathbb{P}_\rho(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^n \mathbb{P}_\rho(y_i|f(\mathbf{x}_i))$ factorising over cases. Finally, after specification of the prior and fitting of the hyperparameters $\theta = \{\phi, \psi, \rho\}$, we wish to compute predictive distributions for test cases.

```
   % 1) SET UP THE GP: COVARIANCE; MEAN, LIKELIHOOD, INFERENCE METHOD
1  mf = {'meanSum',{'meanLinear',@meanConst}};   a = 2; b = 1;    % m(x) = a*x+b
2  cf = {'covSEiso'}; sf = 1; ell = 0.7;  % squared exponential covariance funct
3  lf = 'likLaplace'; sn = 0.2;        % assume Laplace noise with variance sn^2
4  hyp0.mean = [a;b]; hyp0.cov = log([ell;sf]); hyp0.lik  = log(sn); % hypers
5  inf = 'infEP';          % specify expectation propagation as inference method
   % 2) MINIMISE NEGATIVE LOG MARGINAL LIKELIHOOD nlZ wrt. hyp;  do 50 CG steps
6  Ncg = 50; [hyp, nlZ] = minimize(hyp0, 'gp', -Ncg, inf, mf, cf, lf, X, y);
   % 3) PREDICT AT UNKNOWN TEST INPUTS
7  [ymu, ys2] = gp(hyp, inf, mf, cf, lf, X, y, Xs);              % test input Xs
```

In **line 1**, we specify the mean $m_\phi(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} + b$ of the GP with hyperparameters $\phi = \{\mathbf{a}, b\}$. First, the functional form of the mean function is given and its parameters are initialised. The desired mean function, happens not to exist in the library of mean functions; instead we have to make a *composite* mean function from *simple* constituents. This is done using a nested cell array containing the algebraic expression for $m(\mathbf{x})$: As the sum of a linear (`mean/meanLinear.m`) and a constant mean function (`mean/meanConst.m`) it is an affine function. In addition to linear and constant mean functions, the toolbox offers $m(\mathbf{x}) = 0$ and $m(\mathbf{x}) = 1$. These *simple* mean functions can be combined by *composite* mean functions to obtain sums (`mean/meanSum.m`) $m(\mathbf{x}) = \sum_j m_j(\mathbf{x})$, products $m(\mathbf{x}) = \prod_j m_j(\mathbf{x})$, scaled versions $m(\mathbf{x}) = \alpha m_0(\mathbf{x})$ and powers $m(\mathbf{x}) = m_0(\mathbf{x})^d$. This flexible mechanism is used for convenient specification of an extensible algebra of mean functions. Note that functions are referred to either as name strings `'meanConst'` or alternatively function handles `@meanConst`. The order of components of the hyperparameters ϕ is the same as in the specification of the cell array. Every mean function implements its evaluation $\mathbf{m} = m_\phi(\mathbf{X})$ and first derivative computation $\mathbf{m}_i = \frac{\partial}{\partial \phi_i} m_\phi(\mathbf{X})$ on a data set $\mathbf{X}$.

In the same spirit, the squared exponential covariance $k_\psi(\mathbf{x},\mathbf{x}') = \sigma_f{}^2 \exp(-\|\mathbf{x}-\mathbf{x}'\|^2/2\ell^2)$ (`cov/covSEiso.m`) with hyperparameters $\psi = \{\ln \ell, \ln \sigma_f\}$ is set up in **line 2**. Note, that the hyperparameters are represented by the logarithms, as these parameters are naturally positive. Many other *simple* covariance functions are contained in the toolbox. Among others, we offer linear, constant, Matérn, rational quadratic, polynomial, periodic, neural network and finite support covariance functions. *Composite* covariance functions allow for sums $k(\mathbf{x},\mathbf{x}') = \sum_j k_j(\mathbf{x},\mathbf{x}')$, products $k(\mathbf{x},\mathbf{x}') = \prod_j k_j(\mathbf{x},\mathbf{x}')$, positive scaling $k(\mathbf{x},\mathbf{x}') = \sigma_f^2 k_0(\mathbf{x},\mathbf{x}')$ and masking of components $k(\mathbf{x},\mathbf{x}') = k_0(\mathbf{x}_I,\mathbf{x}'_I)$ with $I \subseteq [1,2,..,D]$, $\mathbf{x} \in \mathbb{R}^D$. Again, the interface is simple since only the evaluation of the covariance matrix $\mathbf{K} = k_\psi(\mathbf{X})$ and its derivatives $\partial_i \mathbf{K} = \frac{\partial}{\partial \psi_i} k_\psi(\mathbf{X})$ on a data set $\mathbf{X}$ are required. Furthermore, we need cross terms $\mathbf{k}_* = k_\psi(\mathbf{X},\mathbf{x}_*)$ and $k_{**} = k_\psi(\mathbf{x}_*,\mathbf{x}_*)$ for prediction. There are no restrictions on the composition of both mean and covariance functions—any combination is allowed including nested composition.

The Laplace (`lik/likLaplace.m`) likelihood $\mathbb{P}_{\boldsymbol{\rho}}(y|f) = \exp(-\sqrt{2}/\sigma_n|y-f|)/\sqrt{2}\sigma_n$ with hyperparameters $\boldsymbol{\rho} = \{\ln\sigma_n\}$ is specified in **line 3**. There are only *simple* likelihood functions: Gaussian, Sech-squared, Laplacian and Student's t for ordinary and sparse regression as well as the error and the logistic function for classification. Again, the same inference code is used for any likelihood function. Although the specification of likelihood functions is simple for the user, writing new likelihood functions is slightly more involved as different inference methods require access to different properties; for example, LA requires second derivatives and EP requires derivatives of moments.

All hyperparameters $\boldsymbol{\theta} = \{\phi, \psi, \boldsymbol{\rho}\}$ are stored in a struct `hyp.{mean,cov,lik}`, which is initialised in **line 4**; we select the approximate inference algorithm EP (`inf/infEP.m`) in **line 5**.

We optimise the hyperparameters $\boldsymbol{\theta} \equiv$ `hyp` by calling the CG optimiser (`util/minimize.m`) with initial value $\boldsymbol{\theta}_0 \equiv$ `hyp0` in **line 6** allowing at most $N = 50$ evaluations of the EP approximation to the marginal likelihood $Z_{EP}(\boldsymbol{\theta})$ as done by `gp.m`. Here, $\mathcal{D} = (\mathbf{X}, \mathbf{y}) \equiv$ `(X,y)` is the training data where $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_n\}$ and $\mathbf{y} \in \mathbb{R}^n$. Under the hood, `gp.m` computes in every step a Gaussian posterior approximation and the derivatives $\frac{\partial}{\partial \boldsymbol{\theta}} \ln Z_{EP}(\boldsymbol{\theta})$ of the marginal likelihood by calling EP.

Predictions with optimised hyperparameters are done in **line 7**, where we call `gp.m` with the unseen test inputs $\mathbf{X}_* \equiv$ `Xs` as additional argument. As a result, we obtain the approximate marginal predictive mean $\mathbb{E}[\mathbb{P}(\mathbf{y}_*|\mathcal{D}, \mathbf{X}_*)] \equiv$ `ymu` and the predictive variance $\mathbb{V}[\mathbb{P}(\mathbf{y}_*|\mathcal{D}, \mathbf{X}_*)] \equiv$ `ys2`.

Likelihood \ Inference	Exact	FITC	EP	Laplace	VB	Type, Output Domain	Alternate Name
Gaussian	✓	✓	✓	✓	✓	regression, $\mathbb{R}$	
Sech-squared			✓	✓	✓	regression, $\mathbb{R}$	logistic distribution
Laplacian			✓		✓	regression, $\mathbb{R}$	double exponential
Student's t				✓	✓	regression, $\mathbb{R}$	
Error function			✓	✓	✓	classification, $\{\pm 1\}$	probit regression
Logistic function			✓	✓	✓	classification, $\{\pm 1\}$	logit regression

Table 1: Likelihood $\leftrightarrow$ inference compatibility in the GPML toolbox

Table 1 gives the legal likelihood/inference combinations. Exact inference and the FITC approximation support the Gaussian likelihood only. Variational Bayesian (VB) inference is applicable to all likelihoods. Expectation propagation (EP) for the Student's t likelihood is inherently unstable due to its non-log-concavity. The Laplace approximation (LA) for Laplace likelihoods is not sensible due to the non-differentiable peak of the Laplace likelihood. Special care has been taken for the non-convex optimisation problem imposed by the combination Student's t likelihood and LA.

If the number of training examples is larger than a few thousand, dense matrix computations become too slow. We provide the FITC approximation for regression with Gaussian likelihood where instead of the exact covariance matrix $\mathbf{K}$, a low-rank plus diagonal matrix $\tilde{\mathbf{K}} = \mathbf{Q} + \mathrm{diag}(\mathbf{K} - \mathbf{Q})$ where $\mathbf{Q} = \mathbf{K}_{\mathbf{u}}^{\top}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{u}}$ is used. The matrices $\mathbf{K}_{\mathbf{uu}}$ and $\mathbf{K}_{\mathbf{u}}$ contain covariances and cross-covariances of and between inducing inputs $\mathbf{u}^i$ and data points $\mathbf{x}^j$. Using `inf/infFITC.m` together with any covariance function wrapped into `cov/covFITC.m` makes the computations feasible for large n.

Acknowledgments

Thanks to Ed Snelson for assisting with the FITC approximation.

References

Joaquin Quiñonero Candela and Carl E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(6):1935–1959, 2005.

Mark N. Gibbs and David J. C. MacKay. Variational Gaussian process classifiers. *IEEE Transactions on Neural Networks*, 11(6):1458–1464, 2000.

Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *UAI*, pages 362–369. Morgan Kaufmann, 2001.

Hannes Nickisch and Carl E. Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078, 10 2008.

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006.

Ed Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, 2006.

Christopher K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(20):1342–1351, 1998.

Journal of Machine Learning Research 11 (2010) 3017-3051 Submitted 4/10; Revised 9/10; Published 11/10

Covariance in Unsupervised Learning of Probabilistic Grammars

Shay B. Cohen SCOHEN@CS.CMU.EDU
Noah A. Smith NASMITH@CS.CMU.EDU
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA

Editors: Alex Clark, Dorota Glowacka, Colin de la Higuera, Mark Johnson and John Shawe-Taylor

Abstract

Probabilistic grammars offer great flexibility in modeling discrete sequential data like natural language text. Their symbolic component is amenable to inspection by humans, while their probabilistic component helps resolve ambiguity. They also permit the use of well-understood, general-purpose learning algorithms. There has been an increased interest in using probabilistic grammars in the Bayesian setting. To date, most of the literature has focused on using a Dirichlet prior. The Dirichlet prior has several limitations, including that it cannot directly model covariance between the probabilistic grammar's parameters. Yet, various grammar parameters are expected to be correlated because the elements in language they represent share linguistic properties. In this paper, we suggest an alternative to the Dirichlet prior, a family of logistic normal distributions. We derive an inference algorithm for this family of distributions and experiment with the task of dependency grammar induction, demonstrating performance improvements with our priors on a set of six treebanks in different natural languages. Our covariance framework permits soft parameter tying within grammars and *across* grammars for text in different languages, and we show empirical gains in a novel learning setting using bilingual, non-parallel data.

Keywords: dependency grammar induction, variational inference, logistic normal distribution, Bayesian inference

1. Introduction

One of the motivating applications for grammar induction, or unsupervised grammatical structure discovery, is for the syntactic analysis of text data. Grammar induction, in that case, may lead to the automatic acquisition of linguistic knowledge and the automatic construction of linguistic analyzers for under-studied text domains and languages, without the costly construction of manually annotated corpora. Grammar induction may also shed light on the cognitive process of language acquisition in humans.

When it comes to the problem of grammar induction from natural language data, a fruitful research direction has built on the view of a grammar as a parameterized, generative process explaining the data (Pereira and Schabes, 1992; Carroll and Charniak, 1992; Chen, 1995; Klein and Manning, 2002, 2004, *inter alia*). If the grammar is a probability model, then learning a grammar means selecting a model from a prespecified model *family*. In much prior work, the family is defined as the set of probabilistic grammar for a fixed set of grammar rules, so that grammar learning amounts to *parameter estimation* from incomplete data: sentences in the language are yields of hid-

den derivations from the grammar. Baker (1979) and Lari and Young (1990) describe how dynamic programming (the "inside-outside" algorithm) can be used within an Expectation-Maximization algorithm (Dempster et al., 1977) to estimate the grammar's probabilities from a corpus of text, in the context-free case.

Probabilistic grammars are attractive for several reasons. Like symbolic grammars, they are amenable to inspection by humans, so that it is relatively easy to understand what tendencies the model has captured if the underlying rules are understandable. Unlike purely symbolic grammars, they model frequency and provide a mechanism for reasoning in the face of ambiguity, which is ubiquitous in natural language. Probabilistic grammars can be specialized (e.g., as hidden Markov models for sequential structures) and generalized (e.g., as lexicalized grammars, as synchronous models over tuples of strings, and as grammars in context-sensitive classes). Probabilistic grammars are widely used to build models in natural language processing from *annotated* data, thus allowing easy comparison between unsupervised and supervised techniques. NLP applications of probabilistic grammars and their generalizations include parsing (Collins, 2003; Klein and Manning, 2003; Charniak and Johnson, 2005), machine translation (Wu, 1997; Ding and Palmer, 2005; Chiang, 2005), and question answering (Wang et al., 2007). Probabilistic grammars are probabilistic models, so they permit the use of well-understood methods for learning.

Meanwhile, in machine learning, significant attention has recently been devoted to Bayesian models. The attraction of Bayesian models is that they manage uncertainty in the face of learning from incomplete data, while permitting the use of background knowledge, in the form of a *prior* over models. This prior can be used to inject bias into a model. Such bias can be especially important in cases where the sample size is not large or when the grammar is highly non-identifiable, two scenarios that hold with grammar induction (see Cohen and Smith, 2010, for a discussion of the size of sample required for estimation of probabilistic grammars).

Bayesian methods have been applied to probabilistic grammars in various ways: specifying priors over the parameters of a PCFG (Johnson et al., 2007; Headden et al., 2009) as well as over the *states* in a PCFG (Finkel et al., 2007; Liang et al., 2007), and even over grammatical derivation structures larger than context-free production rules (Johnson et al., 2006; Cohn et al., 2009). The challenge in Bayesian grammar learning is efficiently approximating probabilistic inference. Variational approximations (Johnson, 2007; Kurihara and Sato, 2006) and randomized sampling approximations (Johnson et al., 2006; Goldwater, 2006) are typically applied.

Much of the Bayesian literature and its application to probabilistic grammars has focused on *conjugate priors* in the form of Dirichlet distributions. Conjugate priors were introduced by Raiffa and Schaifer (1961), who gave a desiderata for prior families, including analytical tractability. We argue that the literature has focused on this desideratum only, ignoring expressive power and interpretability. We begin by motivating the modeling of *covariance* among the probabilities of grammar derivation events, and propose the use of logistic normal distributions (Aitchison, 1986; Blei and Lafferty, 2006) over multinomials to build priors over grammars (Section 3). Our motivation relies on the observation that various grammar parameters are expected to be correlated because of the elements in language they represent share linguistic properties. Noting that grammars are built out of a large collection of multinomials, we introduce *shared* logistic normal distributions to allow *arbitrary* covariance among any grammar probabilities. We then describe efficient inference techniques to support decoding and learning with (shared) logistic normal priors over grammars (Section 4), facing the challenge of non-conjugacy of the logistic normal prior to the multinomial family. We experiment with probabilistic dependency grammar induction from data in six lan-

guages, showing how the new approach performs compared to non-Bayesian alternatives as well as more traditional Dirichlet prior-based alternatives (Section 5.1 and Section 5.2). We then demonstrate that the approach can also be effective when learning from *multilingual*, non-parallel text, softly tying parameters across languages (Section 5.4).

The research results in this paper build on work previously reported by Cohen et al. (2008) and Cohen and Smith (2009). Here we provide a more extensive discussion of the techniques, connections to related work, a full derivation of the variational inference algorithms, and a larger set of experiments on more data sets.

2. Probabilistic Grammars

We begin by discussing the general family of probabilistic grammars to which our methods are applicable. A probabilistic grammar defines a probability distribution over a certain kind of structured object (a derivation of the underlying symbolic grammar) explained step-by-step as a stochastic process. HMMs, for example, can be understood as a random walk through a probabilistic finite-state network, with an output symbol sampled at each state. PCFGs generate phrase-structure trees by recursively rewriting nonterminal symbols as sequences of "child" symbols (each itself either a nonterminal symbol or a terminal symbol analogous to the emissions of an HMM). Our experiments will consider a particular family of PCFGs that represent dependency structure (see Section 2.2).

Each step or emission of an HMM and each rewriting operation of a PCFG is conditionally independent of the others given a single structural element (one HMM or PCFG state); this Markov property permits efficient inference over derivations given a string.

In general, a probabilistic grammar defines the joint probability of a string $\mathbf{x}$ and a grammatical derivation $\mathbf{y}$:[1]

$$p(\mathbf{x},\mathbf{y} \mid \theta) = \prod_{k=1}^{K} \prod_{i=1}^{N_k} \theta_{k,i}^{f_{k,i}(\mathbf{x},\mathbf{y})} = \exp \sum_{k=1}^{K} \sum_{i=1}^{N_k} f_{k,i}(\mathbf{x},\mathbf{y}) \log \theta_{k,i}, \tag{1}$$

where $f_{k,i}$ is a function that "counts" the number of times the kth distribution's ith event occurs in the derivation. The parameters θ are a collection of K multinomials $\langle \theta_1, \ldots, \theta_K \rangle$, the kth of which includes N_k competing events. Letting $\theta_k = \langle \theta_{k,1}, \ldots, \theta_{k,N_k} \rangle$, each $\theta_{k,i}$ is a probability, such that

$$\forall k, \forall i, \qquad \theta_{k,i} \geq 0, \tag{2}$$

$$\forall k, \qquad \sum_{i=1}^{N_k} \theta_{k,i} = 1. \tag{3}$$

As is often the case in probabilistic modeling, there are different ways to carve up the random variables. We can think of $\mathbf{x}$ and $\mathbf{y}$ as correlated structure variables (often $\mathbf{x}$ is known if $\mathbf{y}$ is known), or the derivation event counts $\mathbf{f}(\mathbf{x},\mathbf{y}) = \langle f_{k,i}(\mathbf{x},\mathbf{y}) \rangle_{1 \leq k \leq K, 1 \leq i \leq N_k}$ as an integer-vector random variable (useful for variational inference, in Section 4). In this paper, $\mathbf{x}$ is always observable and $\mathbf{y}$ is hidden until we use gold standard data for testing.

Note that there may be many derivations $\mathbf{y}$ for a given string $\mathbf{x}$—perhaps even infinitely many in some kinds of grammars. For HMMs, there are three kinds of multinomials: a starting state multinomial, a transition multinomial per state and an emission multinomial per state. In that case $K = 2s+1$, where s is the number of states. The value of N_k depends on whether the kth multinomial is the starting state multinomial (in which case $N_k = s$), transition multinomial ($N_k = s$) or

1. A table of notation can be found in Appendix A, Table 4, page 3042.

emission multinomial ($N_k = t$, with t being the number of symbols in the HMM). For PCFGs, each multinomial among the K multinomials correspond to a set of N_k context-free rules headed by the same nonterminal. $\theta_{k,i}$ is then the probability of the ith rule for the kth nonterminal.

The field of grammatical inference also includes algorithms and methods for learning the *structure* of a (formal) language generator or grammar (Angluin, 1988; de la Higuera, 2005; Clark and Thollard, 2004; Clark et al., 2008, *inter alia*). This paper is complementary, focusing on the estimation of the *weights* assigned to the grammar's rules. The choice of using a fixed model family corresponds to a choice to work in a statistical parametric setting; extensions to nonparametric settings are possible (Goldwater, 2006; Johnson et al., 2006; Cohen et al., 2010). We focus on grammars which generate dependency structures for derivations. Dependency syntax is a popular representation that has been found useful in a wide range of natural language applications, including machine translation (Lin, 2004; Gimpel and Smith, 2009), question answering (Wang et al., 2007), as well as deeper semantic processing tasks (Johansson and Nugues, 2007; Das et al., 2010). The grammars used in our experiments are extremely permissive, allowing every possible dependency structure for a sentence (see Section 2.2).

2.1 Simple Example: Class-Based Unigram Model

It is helpful to keep in mind a simple model with a relatively small number of parameters such as a class-based unigram model (Brown et al., 1990). Let the observed symbols in $\mathbf{x}$ range over words in some language's vocabulary Γ. Let each word token x_i have an associated word class from a finite set Λ, denoted y_i; the y_i are all hidden. The derivation in this model is the sequence $\langle y_1, \ldots, y_n \rangle$. The probabilistic model consists of two parts:

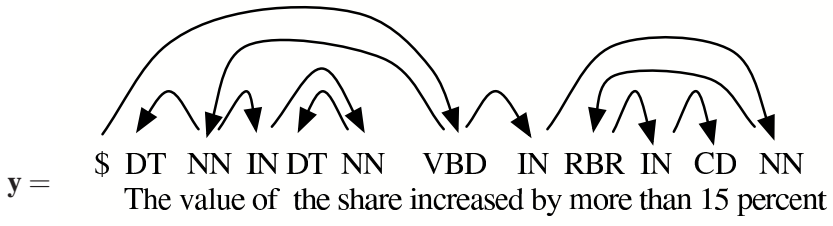
1. For all $y \in \Lambda \cup \{\text{stop}\}$, $\theta_c(y)$ is the probability that the next word will be generated by class y. $\theta_c(\text{stop})$ is the stopping probability.

2. For all $y \in \Lambda$ and all $x \in \Gamma$, $\theta_w(x \mid y)$ is the conditional probability that class y will generate word x.

In this simple model, $K = 1 + |\Lambda|$, $N_1 = |\Lambda|$, and for $k > 1$, $N_k = |\Gamma|$. This model can be thought of as a hidden Markov model with zero order, that is, it has no dependencies between the different hidden states. In addition, if we place a Dirichlet prior on the grammar parameters θ (Section 3.1) and treat θ as a latent variable sampled once per document, this model becomes equivalent to the latent Dirichlet allocation model (Blei et al., 2003). (Our θ_w is denoted β in their notation.) In this case, the derivation vector $\mathbf{y}$ corresponds to a set of topics selected for each word in the bag of words representing the document.

2.2 Dependency Model with Valence

Dependency grammar (Tesnière, 1959) refers to linguistic theories that posit graphical representations of sentences in which words are vertices and the syntax is a directed tree. Such grammars can be context-free or context-sensitive in power, and they can be made probabilistic (Gaifman, 1965). Dependency syntax is used in information extraction, machine translation, question answering, and other natural language processing applications. Our experiments perform unsupervised induction of probabilistic dependency grammars using a model known as "dependency model with valence" (Klein and Manning, 2004). The model is a probabilistic split head automaton grammar



$\mathbf{y} =$

$$\mathbf{x} = \langle \$\ \text{DT NN IN DT NN VBD IN RBR IN CD NN} \rangle$$

$$\begin{aligned}
p(\mathbf{x},\mathbf{y} \mid \theta) &= \theta_c(\text{VBD} \mid \$, \mathsf{r}) \times p(\mathbf{y}^{(1)} \mid \text{VBD}, \theta) \\
p(\mathbf{y}^{(1)} \mid \text{VBD}, \theta) &= \theta_s(\neg\text{stop} \mid \text{VBD}, \mathsf{l}, \mathsf{f}) \times \theta_c(\text{NN} \mid \text{VBD}, \mathsf{l}) \times p(\mathbf{y}^{(2)} \mid \text{NN}, \theta) \\
&\quad \times \theta_s(\text{stop} \mid \text{VBD}, \mathsf{l}, \mathsf{t}) \times \theta_s(\neg\text{stop} \mid \text{VBD}, \mathsf{r}, \mathsf{f}) \times \theta_c(\text{IN} \mid \text{VBD}, \mathsf{r}) \\
&\quad \times p(\mathbf{y}^{(4)} \mid \text{IN}, \theta) \times \theta_s(\text{stop} \mid \text{VBD}, \mathsf{r}, \mathsf{t}) \\
p(\mathbf{y}^{(2)} \mid \text{NN}, \theta) &= \theta_s(\neg\text{stop} \mid \text{NN}, \mathsf{l}, \mathsf{f}) \times \theta_c(\text{DT} \mid \text{NN}, \mathsf{l}) \times \theta_s(\text{stop} \mid \text{DT}, \mathsf{r}, \mathsf{f}) \\
&\quad \times \theta_s(\text{stop} \mid \text{DT}, \mathsf{l}, \mathsf{f}) \theta_c(\text{IN} \mid \text{NN}, \mathsf{r}) \times p(\mathbf{y}^{(3)} \mid \text{IN}, \theta) \\
&\quad \times \theta_s(\text{stop} \mid \text{IN}, \mathsf{l}, \mathsf{f}) \times \theta_s(\text{stop} \mid \text{NN}, \mathsf{l}, \mathsf{t}) \times \theta_s(\text{stop} \mid \text{NN}, \mathsf{r}, \mathsf{t}) \\
p(\mathbf{y}^{(3)} \mid \text{IN}, \theta) &= \theta_s(\neg\text{stop} \mid \text{IN}, \mathsf{r}, \mathsf{f}) \times \theta_c(\text{NN} \mid \text{IN}, \mathsf{r}) \times \theta_c(\text{DT} \mid \text{NN}, \mathsf{l}) \\
&\quad \times \theta_s(\text{stop} \mid \text{DT}, \mathsf{r}, \mathsf{f}) \times \theta_s(\text{stop} \mid \text{DT}, \mathsf{l}, \mathsf{f}) \\
&\quad \times \theta_s(\text{stop} \mid \text{NN}, \mathsf{r}, \mathsf{f}) \times \theta_s(\text{stop} \mid \text{NN}, \mathsf{l}, \mathsf{t}) \\
p(\mathbf{y}^{(4)} \mid \text{IN}, \theta) &= \theta_s(\text{stop} \mid \text{IN}, \mathsf{l}, \mathsf{f}) \times \theta_s(\neg\text{stop} \mid \text{IN}, \mathsf{r}, \mathsf{f}) \times \theta_c(\text{NN} \mid \text{IN}, \mathsf{r}) \\
&\quad \times \theta_s(\text{stop} \mid \text{NN}, \mathsf{r}, \mathsf{f}) \times \theta_s(\neg\text{stop} \mid \text{NN}, \mathsf{l}, \mathsf{f}) \times \theta_c(\text{RBR} \mid \text{NN}, \mathsf{r}) \\
&\quad \times \theta_s(\text{stop} \mid \text{RBR}, \mathsf{l}, \mathsf{f}) \times p(\mathbf{y}^{(5)} \mid \text{RBR}, \theta) \\
p(\mathbf{y}^{(5)} \mid \text{RBR}, \theta) &= \theta_s(\neg\text{stop} \mid \text{RBR}, \mathsf{r}, \mathsf{f}) \times \theta_c(\text{IN} \mid \text{RBR}, \mathsf{r}) \times \theta_c(\text{CD} \mid \text{IN}, \mathsf{r}) \\
&\quad \times \theta_s(\text{stop} \mid \text{IN}, \mathsf{l}, \mathsf{f}) \times \theta_s(\text{stop} \mid \text{IN}, \mathsf{r}, \mathsf{t}) \times \theta_s(\text{stop} \mid \text{CD}, \mathsf{r}, \mathsf{f}) \\
&\quad \times \theta_s(\text{stop} \mid \text{CD}, \mathsf{l}, \mathsf{f})
\end{aligned}$$

Figure 1: An example of a dependency tree (derivation $\mathbf{y}$). and its probability. The part-of-speech tags NN, VBD, DT, CD, RBR, and IN denote noun, past-tense verb, determiner, number, comparative adverb, and preposition, respectively, following Penn Treebank conventions. We break the probability of the tree down into recursive parts, one per head word, marked in blue (lighter). l, r, t, and f denote left, right, true, and false, respectively (see Equation 4).

(Alshawi and Buchsbaum, 1996) that renders inference cubic in the length of the sentence (Eisner, 1997). The language of the grammar is context-free, though our models are permissive and allow the derivation of any string in Γ^*. This is a major point of departure between theoretical work in grammatical inference and work on natural language text, particularly using probabilistic gram-

mars; our goal is to induce a distribution over derivations so that the most likely derivations under the model closely mimic those preferred by linguists (Smith and Eisner, 2005).

"Valence" here refers to the number of arguments controlled by the head of a phrase.[2] In the DMV, each word has a binomial distribution over whether it has at least one left child (similarly on the right), and a geometric distribution over the number of further children (for each side).

Let $\mathbf{x} = \langle x_1, x_2, ..., x_n \rangle$ be a sentence (here, as in prior work, represented as a sequence of part-of-speech tags). x_0 is a special "wall" symbol, \$, on the left of every sentence. A tree $\mathbf{y}$ is defined by a pair of functions $\mathbf{y}_{left}$ and $\mathbf{y}_{right}$ (both $\{0,1,2,...,n\} \rightarrow 2^{\{1,2,...,n\}}$) that map each word to its sets of left and right dependents, respectively. Here, the graph is constrained to be a *projective* tree rooted at $x_0 = \$$: each word except \$ has a single parent, and there are no cycles or crossing dependencies. $\mathbf{y}_{left}(0)$ is taken to be empty, and $\mathbf{y}_{right}(0)$ contains the sentence's single head. Let $\mathbf{y}^{(i)}$ denote the subtree rooted at position i (i.e., $\mathbf{y}^{(i)}$ is a tree consisting of all descendents of x_i in the tree $\mathbf{y}$). The probability $P(\mathbf{y}^{(i)} \mid x_i, \theta)$ of generating this subtree, given its head word x_i, is defined recursively:

$$p(\mathbf{y}^{(i)} \mid x_i, \theta) = \prod_{D \in \{left, right\}} \theta_s(\text{stop} \mid x_i, D, [\mathbf{y}_D(i) = \emptyset]) \times \prod_{j \in \mathbf{y}_D(i)} \theta_s(\neg\text{stop} \mid x_i, D, \text{first}_{\mathbf{y}}(j)) \times \theta_c(x_j \mid x_i, D) \times p(\mathbf{y}^{(j)} \mid x_j, \theta), \quad (4)$$

where $\text{first}_{\mathbf{y}}(j)$ is a predicate defined to be true iff x_j is the closest child (on either side) to its parent x_i. The probability of the entire tree is given by $p(\mathbf{x}, \mathbf{y} \mid \theta) = p(\mathbf{y}^{(0)} \mid \$, \theta)$. The parameters θ are the conditional multinomial distributions $\theta_s(\cdot \mid \cdot, \cdot, \cdot)$ and $\theta_c(\cdot \mid \cdot, \cdot)$. To follow the general setting of Equation 1, we index these distributions as $\theta_1, ..., \theta_K$. Figure 1 shows a dependency tree and its probability under this model (Equation 4).

Note that if all weights θ are greater than zero, the model permits *any* dependency tree over *any* sentence in Γ^*. Hence the goal of grammar induction is to model the *distribution* of derivations, not to separate grammatical strings or derivations from ungrammatical ones.

Klein and Manning's (2004) dependency model with valence is widely recognized as an effective probabilistic grammar for dependency grammar induction. Many recent studies on dependency grammar induction use it. For example, this model has been used to test estimation algorithms such as Viterbi EM (Spitkovsky et al., 2010b), contrastive estimation (Smith and Eisner, 2005), and algorithms which gradually introduce more data to the learning process (Spitkovsky et al., 2010a); it has been used to test the efficacy of multilingual learning through dependency grammar induction (Ganchev et al., 2009; Berg-Kirkpatrick and Klein, 2010); it has been used as a base model that has inspired more complex lexicalized models (Headden et al., 2009). The DMV has also been used as a base model within various estimation techniques with the goal of improving its performance by relying on other properties of language and text such as: dependencies between parameters in the model (Berg-Kirkpatrick et al., 2010), sparsity (Gillenwater et al., 2010), preference for short attachments (Smith and Eisner, 2006), and additional annotation offered by hypertext markup as found on the Web (Spitkovsky et al., 2010c). In addition, the DMV is related to the head-outward model used by Collins (2003) for supervised parsing; Collins' parser is one of the best performing parsers for English. In the rest of the paper, we assume we have a fixed grammar $\mathbf{G}$ for which we estimate the parameters.

2. Here, we refer to "head of a phrase" as in the linguistic sense—the word in a phrase that determines the syntactic category of this phrase.

2.3 Parameter Estimation by Maximum Likelihood

In the original framework, Klein and Manning (2004) treated the DMV as a model on its own, and also in combination with a model over bracketing structures called the "constituent-context model." Here we consider the DMV on its own as it is more capable of generalization and better exemplifies probabilistic grammars.

Klein and Manning learned the DMV using maximum likelihood estimation, carried out by the Expectation-Maximization (EM) algorithm. Because EM for probabilistic grammars has been well documented elsewhere (Lari and Young, 1990; Pereira and Schabes, 1992; Carroll and Charniak, 1992), we only briefly mention that it proceeds by alternating between two steps that update the model parameters. Let $\theta^{(t)}$ denote their values at time step t.

1. E-step: For each training example $\mathbf{x}$, infer the posterior distribution $p(\mathbf{y} \mid \mathbf{x}, \theta^{(t)}, \mathbf{G}) = p(\mathbf{x}, \mathbf{y} \mid \theta^{(t)}) / p(\mathbf{x} \mid \theta^{(t)}, \mathbf{G})$. This is accomplished by dynamic programming (for HMMs, the forward-backward algorithm; for PCFGs, the inside-outside algorithm; for the DMV, an algorithm due to Eisner, 1997), and the result is usually represented as a vector of derivation event expected frequencies, $\langle \mathbb{E}_{p(\cdot \mid \mathbf{x}, \theta^{(t)}, \mathbf{G})} f_{k,i}(\mathbf{x}, \cdot) \rangle_{k,i}$.

2. M-step: Estimate $\theta^{(t+1)}$ from the expected frequencies, as if they were observed frequencies. Since the model is built out of multinomials, there is a closed form solution obtained by normalizing the frequencies.

It is helpful to consider the problem EM iterations aim to solve in its declarative form, the problem of maximizing likelihood:

$$\max_{\theta} p(\mathbf{x} \mid \theta, \mathbf{G}) = \max_{\theta} \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y} \mid \theta, \mathbf{G}).$$

(In fact, EM only locally maximizes this function.) In the above, we suppress the collection of sentences constituting the training data; to be precise, we should take a product of probabilities or a sum of log-probabilities for all training examples:

$$\max_{\theta} \prod_{m=1}^{M} \sum_{\mathbf{y}} p(\mathbf{x}_m, \mathbf{y} \mid \theta, \mathbf{G}).$$

In the Bayesian approach, we treat θ not as a set of parameters to be estimated, but rather as a random event. This contrasts with earlier research that aims to bias the grammar learner with prior information. Klein and Manning (2004), for example, biased the learner by initializing EM with a "harmonic" posterior over dependency attachments that preferred linking words that are closer together in the string to more distant words. Smith and Eisner (2006) more explicitly biased EM by manipulating the posterior calculated in the E-step with penalties for longer dependency attachments or, in an alternative model that permitted disconnected graphs, for contiguity.

3. Bayesian Models over Grammars

An attractive way to incorporate prior knowledge about grammars is through a prior distribution over the grammar's probabilities θ. Priors are often used to obtain *smooth* estimates; Smith (2006)

Model I:

For $m \in \{1, \ldots, M\}$:

1. Draw θ_m from the prior $p(\theta \mid \mathbf{G}, \ldots)$.
2. Draw $(\mathbf{x}_m, \mathbf{y}_m)$ from $p(\mathbf{x}_m, \mathbf{y}_m \mid \theta_m, \mathbf{G})$.

Model II:

1. Draw θ from the prior $p(\theta \mid \mathbf{G}, \ldots)$.
2. For $m \in \{1, \ldots, M\}$:
   Draw $(\mathbf{x}_m, \mathbf{y}_m)$ from $p(\mathbf{x}_m, \mathbf{y}_m \mid \theta, \mathbf{G})$.

Figure 2: Two variations on Bayesian modeling of probabilistic grammars.

explored symmetric Dirichlet priors in the DMV in a maximum *a posteriori* framework for learning that can still be accomplished by EM:

$$\max_{\theta} p(\theta \mid \alpha, \mathbf{G}) \prod_{m=1}^{M} \sum_{\mathbf{y}} p(\mathbf{x}_m, \mathbf{y} \mid \theta, \mathbf{G}), \tag{5}$$

where α denotes the parameters of the prior over grammars. EM is efficient when $p(\theta \mid \alpha, \mathbf{G})$ is a collection of Dirichlet distributions with each $\alpha \geq 1$ (discussed below). For the moment, we leave aside the form of the prior, though it is a major focus of this article.

In this paper, we go farther. We treat θ as a hidden variable, not unlike $\mathbf{y}$. It will therefore be integrated out in defining the probability of the data:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_M \mid \alpha, \mathbf{G}) = \int p(\theta \mid \alpha, \mathbf{G}) \prod_{m=1}^{M} \sum_{\mathbf{y}} p(\mathbf{x}_m, \mathbf{y} \mid \theta, \mathbf{G}) \, d\theta. \tag{6}$$

In this setting, it is α, the distribution over grammar parameters, that encodes knowledge about the grammar, and it will be α that we estimate when we perform learning.

We consider two alternative variations on the Bayesian idea, illustrated in Figure 2. In the first, called "model I," the grammar's probabilities θ are drawn randomly once per sentence for the whole corpus $\mathbf{x}_1, \ldots, \mathbf{x}_M$. In "model II," the grammar parameters are drawn *once* for all of the sentences in the corpus.

Conceptually, both options have advantages and disadvantages when modeling natural language. Drawing θ for each derivation permits more flexibility across derivations, perhaps allowing the learner to capture variation across the corpus (even if not systematically, as the grammars are drawn IID), arising from different authors, for example. Generating θ only once suggests we need to do inference in a smaller space: we only need to find the posterior over a single θ, perhaps leading to better generalization. We will consider both forms in our experiments (Section 5.1).

The question of the choice of a prior distribution still remains. In their pioneering work about conjugate priors,[3] Raiffa and Schaifer (1961) set desiderata for prior distributions in parametric models. These desiderata, which serve as the foundation for *conjugate priors*, include: (i) analytical tractability—the posterior using a certain prior family should stay in the prior family, while it is reasonably easy to identify the posterior from a sample and a prior; (ii) richness—there should be a member in the prior family that is able to express the modeler's beliefs and prior information; (iii)

3. A prior family is conjugate for a family of distributions if the posterior over the family, after observing some data, is also in the prior family. See Raiffa and Schaifer (1961).

interpretability—the prior should be easily interpreted so the modeler can verify that the choice of prior matches prior judgments.

Unfortunately, much of the Bayesian literature for probabilistic grammars and even in general has diverged considerably from these desiderata, and focused only on the first requirement of analytical tractability. As a result, most of the Bayesian language learning literature has focused on Bayesian models with a Dirichlet prior (Johnson et al., 2007; Goldwater and Griffiths, 2007; Toutanova and Johnson, 2007; Kurihara and Sato, 2006, *inter alia*), which is conjugate to the multinomial family. We argue that the last two requirements are actually more important than the first one, which is motivated by mere mathematical and computational convenience. We suggest replacing the first requirement with "computational tractability"—it should be easy to represent the posterior (or an approximation of it) computationally. In that case, the modeler can focus on choosing *rich* priors that can more properly model different structural elements of a grammar. To solve the problem of inference, we can now use approximate inference algorithms such as the one we give in Section 4 and Appendix B. Indeed, approximations are sometimes required even for the conjugate case, and are always required when the data are incomplete.

We next give an overview of the Dirichlet prior that provides analytical tractability for probabilistic grammars, and then demonstrate the alternative which focuses on the second and third requirements, the logistic normal distribution. The logistic normal, we suggest, improves over the Dirichlet from the perspective of desideratum (ii), though we must take further steps to achieve sufficient "richness" to account for arbitrary covariance and for multilingual text data.

3.1 Dirichlet Distributions

From the computational perspective, the Dirichlet distribution is indeed a natural choice for a prior over the parameters of the grammar because of its analytical tractability, which makes inference more elegant and less computationally intensive in both the maximum *a posteriori* (Equation 5) and Bayesian (Equation 6) settings. In addition, a Dirichlet prior can encourage sparse solutions (i.e., many $\theta_{k,i} = 0$), a property which is desirable in natural language learning (Johnson et al., 2007), as it corresponds to eliminating unnecessary grammar rules. (Indeed, learning to exclude rules by setting their probabilities to zero is one way of going about symbolic grammar induction.)

If we use a Dirichlet distribution with a probabilistic grammar, then the hyperparameters for the grammar consist of K vectors with positive elements, the kth of which has length N_k. We denote these hyperparameters by α, in which case the prior over the grammar parameters θ has the form:

$$p(\theta \mid \alpha) = \prod_{k=1}^{K} \left(\frac{\prod_{i=1}^{N_k} \Gamma(\alpha_{k,i})}{\Gamma\left(\sum_{i=1}^{N_k} \alpha_{k,i}\right)} \prod_{i=1}^{N_k} \theta_{k,i}^{\alpha_{k,i}-1} \right) = B(\theta) \times \prod_{k=1}^{K} \prod_{i=1}^{N_k} \theta_{k,i}^{\alpha_{k,i}-1},$$

where $\Gamma(\cdot)$ is the Gamma function and $B(\theta)$ is a constant term with respect to θ.

Consider again the simple model of Section 2.1. If we embed it inside model I (Figure 2) we arrive exactly at the latent Dirichlet allocation model of Blei et al. (2003), where each example is a document (not a sentence).

The Dirichlet distribution can also be derived as a normalized set of variables of exponentiated *independent* Gamma-distributed variables. More precisely, for each multinomial θ_k ($k \in \{1, \ldots, K\}$), we can draw N_k independent random samples $v_{k,1}, \ldots, v_{k,N_k}$ from Gamma distributions with shapes

$\alpha_{k,1},\ldots,\alpha_{k,N_k}$, respectively, and scale 1 and then let:

$$\theta_{k,i} = \frac{v_{k,i}}{\sum_{i'=1}^{N_k} v_{k,i'}}.$$

This alternative representation of the Dirichlet distribution points to a weakness: there is no explicit covariance structure present when θ are drawn from a Dirichlet. The only way θ_k covary is through the normalization that maps $v_{k,i}$ to the probability simplex. In fact, the correlation between $\theta_{k,i}$ and $\theta_{k,i'}$ is always negative and equals $-\frac{(\alpha_{k,i}\alpha_{k,i'})^{1/2}}{((\alpha_{k,0}-\alpha_{k,i})(\alpha_{k,0}-\alpha_{k,i'}))^{1/2}}$ where $\alpha_{k,0} = \sum_{i=1}^{N_k} \alpha_{k,i}$. This relates back to the desiderata of Raiffa and Schaifer: the covariance (and in fact, variance) structure that the Dirichlet distribution offers is not rich. This is especially true when modeling language, as we explain in the section below.

3.2 Modeling Covariance with Logistic Normal Distributions

When we consider probabilistic grammars for natural languages, especially those over words or word classes like parts of speech, we *do* expect to see covariance structure. Intuitively, the probability of a particular word or word class having singular nouns as arguments is likely tied to the probability of the same word having *plural* nouns as arguments. Words that tend to attach to one type of parent are expected to tend to attach to similar parents. This follows because words and word classes tend to follow patterns. This is a large part of the empirical motivation for syntactic theories that make use of part of speech and phrase categories.

A natural candidate for a distribution that models covariance is the multivariate normal distribution. However, values drawn from the multivariate normal distribution can be both positive and negative, and they also do not necessarily normalize to 1, both are requirements from θ (see Equations 2–3). Aitchison (1986) suggested a logistic transformation on a multivariate normal variable to get values which correspond to points on the probabilistic simplex. He called it the "logistic normal" distribution.

The logistic normal (LN) distribution maps a $(d-1)$-dimensional multivariate Gaussian to a distribution on the d-dimensional probability simplex, $\{\langle z_1,\ldots,z_d\rangle \in \mathbb{R}^d : z_i \geq 0, \sum_{i=1}^d z_i = 1\}$, as follows:

1. Draw $\eta = \langle \eta_1,\ldots,\eta_{d-1}\rangle$ from a multivariate Gaussian with mean μ and covariance matrix Σ.
2. Let $\eta_d = 0$.
3. For $i \in \{1,\ldots,d\}$, let:
$$z_i = \frac{\exp \eta_i}{\sum_{j=1}^d \exp \eta_j}.$$

Drawing from a $(d-1)$-dimensional Gaussian preserves identifiability; a d-dimensional Gaussian would have an extra degree of freedom, allowing more than one outcome of η to lead to the same $\mathbf{z}$.

For probabilistic grammars, we define one LN distribution per multinomial. This gives a prior over each θ_k that permits covariance among $\langle \theta_{k,1},\ldots,\theta_{k,N_k}\rangle$.

Blei and Lafferty (2006) and Ahmed and Xing (2007) successfully used the LN distribution for topic models, extending the latent Dirichlet allocation model (Blei et al., 2003). In Cohen et al.

$$
\left.\begin{array}{rcccccccc}
I_1 & = & \{1{:}2, 3{:}6, 7{:}9\} & = & \{ & I_{1,1}, & I_{1,2}, & I_{1,L_1} & \} \\
I_2 & = & \{1{:}2, 3{:}6\} & = & \{ & I_{2,1}, & I_{2,L_2} & & \} \\
I_3 & = & \{1{:}4, 5{:}7\} & = & \{ & & I_{3,1}, & I_{3,L_3} & \} \\
I_N & = & \{1{:}2\} & = & \{ & I_{4,L_4} & & & \} \\
 & & & & & J_1 & J_2 & J_K &
\end{array}\right\} \text{prt. struct. } \mathcal{S}
$$

$$
\left.\begin{array}{rcll}
\eta_1 & = & \langle \eta_{1,1}, \eta_{1,2},\ \eta_{1,3}, \eta_{1,4}, \eta_{1,5}, \eta_{1,6},\ \eta_{1,7}, \eta_{1,8}, \eta_{1,\ell_1} \rangle & \sim \text{Normal}(\mu_1, \Sigma_1) \\
\eta_2 & = & \langle \eta_{2,1}, \eta_{2,2},\ \eta_{2,3}, \eta_{2,4}, \eta_{2,5}, \eta_{2,\ell_2} \rangle & \sim \text{Normal}(\mu_2, \Sigma_2) \\
\eta_3 & = & \langle \eta_{3,1}, \eta_{3,2}, \eta_{3,3}, \eta_{3,4},\ \eta_{3,5}, \eta_{3,6}, \eta_{3,\ell_3} \rangle & \sim \text{Normal}(\mu_3, \Sigma_3) \\
\eta_4 & = & \langle \eta_{4,1}, \eta_{4,\ell_4} \rangle & \sim \text{Normal}(\mu_4, \Sigma_4)
\end{array}\right\} \text{sample } \eta
$$

$$
\left.\begin{array}{rcl}
\tilde{\eta}_1 & = & \frac{1}{3}\langle \eta_{1,1} + \eta_{2,1} + \eta_{4,1},\ \eta_{1,2} + \eta_{2,2} + \eta_{4,2} \rangle \\
\tilde{\eta}_2 & = & \frac{1}{3}\langle \eta_{1,3} + \eta_{2,3} + \eta_{3,1},\ \eta_{1,4} + \eta_{2,4} + \eta_{3,2},\ \eta_{1,5} + \eta_{2,5} + \eta_{3,3}, \\
 & & \eta_{1,6} + \eta_{2,6} + \eta_{3,4} \rangle \\
\tilde{\eta}_3 & = & \frac{1}{2}\langle \eta_{1,7} + \eta_{3,5},\ \eta_{1,8} + \eta_{3,6},\ \eta_{1,9} + \eta_{3,7} \rangle
\end{array}\right\} \text{combine } \eta
$$

$$
\left.\begin{array}{rcl}
\theta_1 & = & (\exp \tilde{\eta}_1) \Big/ \sum_{i'=1}^{N_1} \exp \tilde{\eta}_{1,i'} \\
\theta_2 & = & (\exp \tilde{\eta}_2) \Big/ \sum_{i'=1}^{N_2} \exp \tilde{\eta}_{2,i'} \\
\theta_3 & = & (\exp \tilde{\eta}_3) \Big/ \sum_{i'=1}^{N_3} \exp \tilde{\eta}_{3,i'}
\end{array}\right\} \text{softmax}
$$

Figure 3: An example of a shared logistic normal distribution, illustrating Def. 1. $N = 4$ experts are used to sample $K = 3$ multinomials; $L_1 = 3, L_2 = 2, L_3 = 2, L_4 = 1, \ell_1 = 9, \ell_2 = 6, \ell_3 = 7, \ell_4 = 2, N_1 = 2, N_2 = 4$, and $N_3 = 3$. From top to bottom: the partition structure $\mathcal{S}$ describes I_j which tell how segment a normal expert into parts which are matched to multinomials ("prt. struct. $\mathcal{S}$"). Each normal expert is sampled from a multivariate normal ("sample η"), and then matched and averaged according to the partition strcture ("combine η"). The final step is exponentiating and normalizing η to get θ ("softmax"). This figure is best viewed in color.

(2008), we demonstrated how the LN distribution is an effective alternative to the Dirichlet for probabilistic dependency grammar induction in the Bayesian setting.

We note that the family of logistic normal distributions and the family of Dirichlet distributions are very different from each another. One cannot find two distributions from each class which are arbitrary close to each other in any meaningful sense. However, it can be shown (Aitchison, 1986) that given a Dirichlet distribution with very large α, we can find a logistic normal distribution such that the KL-divergence between the Dirichlet distribution and logistic normal distribution is small.

3.3 Sharing Across Multinomials

The LN distribution has an inherent limitation when we consider probabilistic models made up of more than one multinomial distribution, such as probabilistic grammars. Each multinomial is drawn separately from an independent Gaussian, so that covariance can only be imposed among events competing within one multinomial, not across multinomials. With the DMV, for example, the probability of a past-tense verb (VBD) having a noun as a right child might correlate with the

probability that other kinds of verbs (VBZ, VBN, etc.) have a noun as a right child. This correlation cannot be captured by the LN distribution, because the VBZ and VBN as parents are represented using their own multinomials over children, unrelated to that of VBD as a parent.

One way to mend this limitation is to define a single Gaussian over $N \triangleq \sum_{k=1}^{K} N_k$ variables with one $N \times N$ covariance matrix. Then, instead of applying the logistic transformation to the whole vector as a single multinomial, we can apply it to subvectors to get disjoint multinomials. When learning, the large covariance matrix captures correlations between all pairs of events in all multinomials. The induced distribution is called the *partitioned* logistic normal (PLN) distribution. It is a generalization of the LN distribution (see Aitchison, 1986).

In practice, creating a covariance matrix of size $N \times N$ is likely to be too expensive. DMV, for example, has $O(t^2)$ weights for a part-of-speech vocabulary of size t, requiring a very large multivariate normal distribution with $O(t^4)$ covariance parameters.

To solve this problem, we suggest a refinement of the class of PLN distributions. Instead of using a single normal vector for all of the multinomials, we use several normal vectors, partition each one and then *recombine* parts which correspond to the same multinomial, as an average. Next, we apply the logistic transformation on the mixed vectors (each of which is normally distributed as well). Figure 3 gives an example of a non-trivial case of using a SLN distribution, where three multinomials are generated from four normal experts.

We now formalize this notion. For a natural number N, we denote by $1{:}N$ the set $\{1,\ldots,N\}$. For a vector in $v \in \mathbb{R}^N$ and a set $I \subseteq 1{:}N$, we denote by v_I the vector created from v by using the coordinates in I. Recall that K is the number of multinomials in the probabilistic grammar, and N_k is the number of events in the kth multinomial. We define a shared logistic normal distribution with N "experts" over a collection of K multinomial distributions:

Definition 1 *Let $\eta_n \sim \text{Normal}(\mu_n, \Sigma_n)$ be a set of multivariate normal variables for $n \in 1{:}N$, where the length of η_n is denoted ℓ_n. Let $I_n = \{I_{n,j}\}_{j=1}^{L_n}$ be a partition of $1{:}\ell_n$ into L_n sets, such that $\cup_{j=1}^{L_n} I_{n,j} = 1{:}\ell_n$ and $I_{n,j} \cap I_{n,j'} = \emptyset$ for $j \neq j'$. Let J_k for $k \in 1{:}K$ be a collection of (disjoint) subsets of $\{I_{n,j} \mid n \in 1{:}N, j \in 1{:}\ell_n, |I_{n,j}| = N_k\}$, such that all sets in J_k are of the same size, N_k. Let $\tilde{\eta}_k = \frac{1}{|J_k|} \sum_{I_{n,j} \in J_k} \eta_{n,I_{n,j}}$, and $\theta_{k,i} = \exp(\tilde{\eta}_{k,i}) \big/ \sum_{i'} \exp(\tilde{\eta}_{k,i'})$. We then say θ distributes according to the* shared logistic normal distribution *with* partition structure $\mathcal{S} = (\{I_n\}_{n=1}^{N}, \{J_k\}_{k=1}^{K})$ *and* normal experts $\{(\mu_n, \Sigma_n)\}_{n=1}^{N}$ *and denote it by* $\theta \sim \text{SLN}(\mu, \Sigma, \mathcal{S})$.

The partitioned LN distribution in Aitchison (1986) can be formulated as a shared LN distribution where $N = 1$. The LN collection presented in Section 3.2 is the special case where $N = K$, each $L_n = 1$, each $\ell_k = N_k$, and each $J_k = \{I_{k,1}\}$.

We note that there is an issue with identifiability that we need to resolve with SLN distributions, as with the LN distribution. It is required that for all multinomials, we set the first value of the samples from the normal expert to 0. For simplicity, we did not include it explicitly in Definition 1, because this can be achieved by setting the normal expert's mean and variance values to 0 in the first index of each normal expert ($\eta_{n,1} = 0$ for all n).

The covariance among arbitrary $\theta_{k,i}$ is not defined directly; it is implied by the definition of the normal experts $\eta_{n,I_{n,j}}$, for each $I_{n,j} \in J_k$. We note that a SLN can be represented as a PLN by relying on the distributivity of the covariance operator, and merging all the partition structure into one (perhaps sparse) covariance matrix. SLNs, in that case, represent a subset of PLNs with a factored structure on the covariance matrices.

It is convenient to think of each $\eta_{i,j}$ as a weight associated with a unique event's probability, a certain outcome of a certain multinomial in the probabilistic grammar. By letting different $\eta_{i,j}$ covary with each other, we strengthen the relationships among $\theta_{k,j}$ and permit learning of the one to affect the learning of the other. Definition 1 also implies that we multiply several multinomials together in a product-of-experts style (Hinton, 1999), because the exponential of an average of normals becomes a product of (unnormalized) probabilities.

We note that the partition structure is a hyperparameter. In our experiments, it encodes domain knowledge about the languages we experiment with (Section 5.3). We believe this is a key advantage of SLN in this setting: marrying the notions of prior knowledge and a Bayesian prior. The beliefs of the model about a language can be encoded into a distribution over the parameters. We leave for future work the discovery of partition structure during the learning process.

3.4 Local Log-Linear Models over Parameters

We give now another interpretation of the shared logistic normal prior using a feature representation, which is related to recent work by Berg-Kirkpatrick et al. (2010). A probabilistic grammar with a shared logistic normal prior can be thought of as a probabilistic grammar where the grammar's parameters are themselves modeled using a local log-linear model with a Gaussian prior over the weights of this log-linear model. Let θ_k be a multinomial in the collection of multinomials for a probabilistic grammar. Then, according to Definition 1 we have:

$$\theta_{k,i} = \frac{\exp\left(g_k(i) \cdot \eta\right)}{Z_k(\eta)},$$

where η is a vector of length $\sum_{n=1}^{N} \ell_n$, a concatenation of all normal experts, and $g_k(i)$ is a feature vector, again of length $\sum_{n=1}^{N} \ell_n$, which is divided into subvectors $g_{k,n}(i)$ each of length ℓ_n. $g_{k,n,j}(i) = 1/|J_k|$ if the ith event in the kth multinomial uses the jth coordinate of the nth normal expert—that is, there exists an $I_{n,r} \in I_n \cap J_k$ such that $j \in I_{n,r}$ (according to Definition 1)—and 0 otherwise. The term $Z_k(\eta)$ is a normalization constant of the form:

$$Z_k(\eta) = \sum_{i'} \exp\left(g_k(i') \cdot \eta\right).$$

Note that the features in the local log-linear model refer to the *hyperparameters* of the SLN, more specifically, the partition structure. They do not refer to the observed data or the latent structural elements in the probabilistic grammar. These features have a Gaussian prior over them, represented by the normal experts' mean values and covariance matrices (μ and Σ). In that case, the Gaussian prior which we optimize during inference using empirical Bayes (Section 4) can be thought of as a quadratic penalty on the local log-linear weights. We note that in most cases in the literature, Gaussian priors (or L_2 regularizers) are used with mean value 0 and a uniform diagonal covariance matrix, in order to push feature weights to values close to 0. This is not the case with our model.

Berg-Kirkpatrick et al. (2010) used the idea of local log-linear models for several natural language processing tasks, including dependency grammar induction and part-of-speech tagging. Instead of using features that are based on a Gaussian prior, they used a set of ordinary binary features, which describe relationships between different parameters in a similar way to the ones presented in Section 5.3.

4. Inference and Learning

Having defined a family of probability models over grammars, we now consider the problem of inferring posterior distributions under this model. We first consider inference over $\mathbf{y}$, then over θ, then learning the parameters of the distribution over grammars in an empirical Bayesian framework.

4.1 Decoding: Inferring y

Classical statistical approaches to language processing normally assume that inputs (here, sentences $\mathbf{x}$) are independently and identically distributed. Decoding is the problem of choosing an analysis (here, grammatical derivation $\mathbf{y}$) given the input. Most commonly this is accomplished by choosing the most probable analysis:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}, \theta, \mathbf{G}) = \operatorname*{argmax}_{\mathbf{y}} p(\mathbf{x}, \mathbf{y} \mid \theta, \mathbf{G}). \tag{7}$$

This is commonly called "Viterbi" decoding, referring to the algorithm that accomplishes the maximization for hidden Markov models. An alternative is to choose the analysis that minimizes *risk*, or the expectation (under the model) of a cost function. Let $\text{cost}(\mathbf{y}, \mathbf{y}^*)$ denote the nonnegative cost of choosing analysis $\mathbf{y}$ when the correct analysis is $\mathbf{y}^*$.

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}} \mathbb{E}_{p(\cdot \mid \mathbf{x}, \theta, \mathbf{G})} \text{cost}(\mathbf{y}, \cdot) = \operatorname*{argmax}_{\mathbf{y}} \sum_{\mathbf{y}'} p(\mathbf{y}' \mid \mathbf{x}, \theta, \mathbf{G}) \ \text{cost}(\mathbf{y}, \mathbf{y}').$$

This is known as minimum Bayes risk (MBR) decoding.[4] For dependency parsing, the cost function counts the number of words attached to the wrong parent.

Decoding is a crucial step in evaluation of models of natural language. Typically for supervised and unsupervised models, decoding output is compared to expert human-annotated gold standard analyses, providing an objective measure of the quality of the learned model. Best practice measures quality on new test data unseen during training, to test the generalization ability of the learned model. This is an attractive approach to evaluating the quality of unsupervisedly induced grammars.

In the Bayesian setting, decoding might be accomplished using the posterior over derivations, marginalizing out the unknown grammar weights. For model I, Viterbi decoding would correspond to:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}} p(\mathbf{y} \mid \alpha, \mathbf{G}) = \operatorname*{argmax}_{\mathbf{y}} \int p(\theta \mid \alpha, \mathbf{G}) p(\mathbf{x}, \mathbf{y} \mid \theta, \mathbf{G}) \, d\theta. \tag{8}$$

Unfortunately, there is no closed-form solution for the integral in Equation 8 and finding $\mathbf{y}^*$ is intractable. We therefore have to resort to approximate inference (Section 4.2). Model II creates dependence among the derivations of the different sentences in the training set, requiring a different inference procedure.

In this work, we consider three decoding techniques. The first takes a point estimate of θ and applies Viterbi decoding (Equation 7). The point estimate is derived using techniques discussed below. After estimating the μ (and the Σ), we use the logistic transformation on μ to obtain this point estimate for Viterbi decoding. Recall that for the DMV, decoding can be accomplished in cubic time using dynamic programming (Section 2.2).

4. In some cases, decoding selects only certain salient aspects of a derivation, such as the derived tree corresponding to a tree adjoining grammar's *derivation* tree. In such cases, Viterbi and/or MBR decoding may require approximations.

The second approach makes use of the same point estimate of θ, only with MBR decoding, as described above. The loss function we use is dependency attachment error, for the task of dependency grammar induction. MBR decoding in this case works as follows: using θ and the inside-outside algorithm, we compute the posterior probability of each dependency attachment (directed edge in the graph) being present in the grammatical derivation for the sentence. Then, we find the tree with the largest score, the score being the sum of the posterior probabilities of each edge present in the tree.

Neither Viterbi nor MBR decoding uses the entire distribution over grammar weights. In the LN case, for example, the covariance matrix Σ is ignored. We suggest "committee decoding," in which a set of randomly sampled grammar weights are drawn for each sentence to be parsed. The weights are drawn from the learned distribution over grammar weights, parameterized by μ and Σ in the LN case. Viterbi or MBR decoding can then be applied. Note that this decoding mechanism is randomized: we sample a grammar per sentence, and use it to decode. We apply this decoding mechanism ten times, and average performance. This decoding method is attractive because it has generalization error guarantees: in a PAC-Bayesian framework, it can be shown that the error of committee parsing on the sample given should be close to the expected error (see Seeger, 2002; McAllester, 2003; Banerjee, 2006).

4.2 Variational Inference with Logistic Normal Distributions

The lack of conjugacy of the logistic normal distribution to the multinomial family complicates the inference of distributions over θ and distributions over the hidden derivations $\mathbf{y}$ from the probabilistic grammar, given a sequence of observed sentences $x_1, \ldots, x_M$.

Mimno et al. (2008) explored inference with the logistic normal distribution using sampling with an auxiliary variable method. However, sampling is notoriously slow to converge, especially with complicated structures such as grammatical derivations. The algorithm Mimno et al. suggest is also rather complicated, while alternatives, such as mean-field variational inference (Wainwright and Jordan, 2008), offer faster convergence and a more intuitive solution to the problem of non-conjugacy of the logistic normal distribution.

Variational inference algorithms have been successfully applied to various grammar and syntax learning tasks (Kurihara and Sato, 2006; Liang et al., 2007; Headden et al., 2009; Boyd-Graber and Blei, 2010; Cohen et al., 2010, *inter alia*). We give the full technical details of mean-field variational inference for probabilistic grammars with logistic normal priors in Appendix B, and turn to give a brief overview of the main technical details next, under the simplifying assumption that we have a single observation $\mathbf{x}$.

Mean-field variational inference in the Bayesian setting relies on two principal approximations: the first approximation is done to the marginalized log-likelihood. Using Jensen's inequality and an auxiliary distribution $q(\theta, \mathbf{y})$, later to be used as our approximate posterior, we bound the log-likelihood, marginalizing out the parameters and the hidden derivations in the grammar:

$$\log \int \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}, \theta \mid \mu, \Sigma, \delta, \mathbf{G}) \, d\theta \geq \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{y}, \theta \mid \mu, \Sigma, \delta, \mathbf{G})]. \tag{9}$$

The goal of the approximation in Equation 9 is to derive a bound which is optimized with respect to q, instead of optimizing the marginalized log-likelihood, which is intractable. q serves as our approximate posterior.

The bound in Equation 9 requires further approximation, the mean-field approximation, to be tractable. This mean-field approximation states that $q(\theta, \mathbf{y})$ is factorized and has the following form:

$$q(\theta, \mathbf{y}) = q(\theta)q(\mathbf{y}).$$

The variational distributions, $q(\theta)$ and $q(\mathbf{y})$ can take an arbitrary form, as long as the bound in Equation 9 can be efficiently maximized with respect to these variational distributions. For the case of logistic normal priors, an additional approximation will be necessary (a first-order Taylor approximation to the log of the normalization of the logistic normal distribution), because of the lack of conjugacy of the logistic normal priors to the multinomial family (see Appendix B). We show in Appendix B that even though $q(\mathbf{y})$ can have an arbitrary form, in order to maximize the variational bound it needs to have the form of a probabilistic grammar, dominated by the grammar's variational parameters. This makes inference applicable through the use of an inside-outside algorithm with a weighted grammar of the same form as the original model. The mean-field approximation yields an elegant algorithm, which looks similar to the Expectation-Maximization algorithm (Section 2.3), alternating between optimizing the bound in Equation 9 with respect to $q(\theta)$ and with respect to $q(\mathbf{y})$.

4.3 Variational EM

The variational inference algorithm in Section 4.2 assumes that the μ and Σ are fixed. We are interested in obtaining an *estimate* for μ and Σ, so that we can fit the data and then use the learned model as described in Section 4.1 to decode new data (e.g., the test set in our experiments). To achieve this, we will use the above variational method within an EM algorithm that estimates μ and Σ in empirical Bayes fashion. (For Viterbi and MBR decoding, we then estimate θ as μ, the mean of the learned prior; see Section 4.1.) In the E-step, we maximize the bound with respect to the variational parameters using coordinate ascent as in Section 4.2. We optimize each of these separately in turn, cycling through them, using appropriate optimization algorithms for each. In the M-step, we apply maximum likelihood estimation with respect to μ and Σ given sufficient statistics gathered from the variational parameters in the E-step. Appendix C describes the algorithm in full.

5. Experiments

We applied our modeling framework to unsupervised learning of the dependency model discussed in Section 2.2. We consider four scenarios:

1. (Section 5.1) Experiments with dependency grammar induction for English text using the logistic normal distribution.
2. (Section 5.2) Experiments with text in five additional languages: Chinese, Portuguese, Turkish, Czech, and Japanese.
3. (Section 5.3) Experiments with the shared logistic normal distribution for tying parameters which correspond to the same coarse part-of-speech tag (English, Portuguese, and Turkish).
4. (Section 5.4) Experiments with the shared logistic normal distribution in *bilingual* settings (English, Portuguese, and Turkish).

	attachment accuracy (%)								
	Viterbi decoding			MBR decoding			Committee decoding		
	≤ 10	≤ 20	all	≤ 10	≤ 20	all	≤ 10	≤ 20	all
MLE	45.8	39.1	34.2	46.1	39.9	35.9	*		
Dirichlet-I	45.9	39.4	34.9	46.1	40.6	36.9	*		
LN-I, $\Sigma_k^{(0)} = \mathbf{I}$	56.5	42.9	36.6	58.4	45.2	39.5	**56.4**$_{\pm.001}$	**42.3**$_{\pm.001}$	**36.2**$_{\pm.001}$
LN-I, families	**59.3**	**45.1**	**39.0**	**59.4**	**45.9**	**40.5**	56.3$_{\pm.01}$	41.3$_{\pm.01}$	34.9$_{\pm.005}$
LN-II, $\Sigma_k^{(0)} = \mathbf{I}$	26.1	24.0	22.8	27.9	26.1	25.3	22.0$_{\pm.02}$	20.1$_{\pm.02}$	19.1$_{\pm.02}$
LN-II, families	24.9	21.0	19.2	26.3	22.8	21.5	26.6$_{\pm.003}$	22.7$_{\pm.003}$	20.8$_{\pm.0006}$

Table 1: Attachment accuracy of different learning methods on unseen test data from the Penn Treebank of varying levels of difficulty imposed through a length filter. MLE is a reproduction of an earlier result using EM (Klein and Manning, 2004). LN-I and LN-II denote using the logistic normal with model I and model II (Figure 2), respectively. Committee decoding includes ten averaged runs. Numbers in small font denote variance. Results in bold denote best results in a column. Training is done on sentences of length ≤ 10, though testing is done on longer sentences as well.

5.1 English Text

We begin our experiments with the *Wall Street Journal* Penn treebank (Marcus et al., 1993). Following standard practice, sentences were stripped of words and punctuation, leaving part-of-speech tags for the unsupervised induction of dependency structure. We note that, in this setting, using gold standard part-of-speech tags as the input to the learning algorithm is common (Klein and Manning, 2004; Smith and Eisner, 2006; Spitkovsky et al., 2010b,a; Gillenwater et al., 2010, *inter alia*).

We train on §2–21, tune on §22 (without using annotations), and report final results on §23. Details of this data set (and others) are found in Table 2. Unsupervised training for these data sets can be costly, and requires iteratively running a cubic-time inside-outside dynamic programming algorithm, so we follow Klein and Manning (2004) in restricting the training set to sentences of ten or fewer words in length. Short sentences are also less structurally ambiguous and may therefore be easier to learn from.

To evaluate the performance of our models, we report the fraction of words whose predicted parent matches the gold standard annotation in the treebank.[5] This performance measure is known as *attachment accuracy*. We will report attachment accuracy on three subsets of the test corpus: sentences of length ≤ 10 (typically reported in prior work and most similar to the training data set), length ≤ 20, and the full test corpus. We considered the three decoding methods mentioned in Section 4.1. For MBR decoding, we use the number of dependency attachment errors as the loss function. This means that at decoding time, we minimize the expected number of attachment errors according to the prediction of the estimated model. Because committee decoding is a randomized algorithm, we run it ten times on the unseen data, and then average the dependency attachment accuracy.

5. The Penn Treebank's phrase-structure annotations were converted to dependencies using the head rules of Yamada and Matsumoto, which are very similar to the ones by Collins (1999). See `http://www.jaist.ac.jp/~h-yamada`.

Initialization is important for all conditions, because likelihood and our variational bound are non-concave functions. For the values of the multinomials (θ), we use the harmonic initializer from Klein and Manning (2004). It estimates θ using soft counts on the training data where, in an n-length sentence, (i) each word is counted as the sentence's head $\frac{1}{n}$ times, and (ii) each word x_i attaches to x_j proportional to $|i-j|^{-1}$, normalized to a single attachment per word. This initializer is used with MLE and Dirichlet-I ("I" stands for model I from Figure 2). In the case of LN-I and LN-II, it is used as an initializer both for μ and inside the E-step.

For learning with the logistic normal prior, we consider two initializations of the covariance matrices Σ_k. The first is the $N_k \times N_k$ identity matrix. We then tried to bias the solution by injecting prior knowledge about the part-of-speech tags. To do that, we manually mapped the tag set (34 tags) to twelve disjoint tag "families." These are simply coarser tags: adjective, adverb, conjunction, foreign, interjection, noun, number, particle, preposition, pronoun, proper, verb. The coarse tags were chosen to loosely account for the part-of-speech tag sets of seven treebanks in different languages. The mapping from fine-grained tags to coarser tags are based on the annotation guidelines of the relevant treebank. This mapping into families provides the basis for an initialization of the covariance matrices for the dependency distributions: 1 on the diagonal, 0.5 between probabilities of possible child tags that belong to the same family, and 0 elsewhere. These results are denoted "families" and are compared to the identity matrix as an initializer.

We compared several models, where learning is accomplished using (variational) EM: MLE, standard maximum-likelihood estimation using EM; Dirichlet-I, a common baseline in the literature which uses a Dirichlet prior together with variational EM; and LN-I (LN-II), a model with the logistic normal distribution using model I (model II). In all cases, we either run the (variational) EM algorithm until convergence of the log-likelihood (or its bound) or until the log-likelihood on an unannotated development set of sentences does not increase.

We note that on the full test set, attaching each word to the word on its right ("Attach-Right") achieves about 30% accuracy, and attaching each word to the word on its left ("Attach-Left") achieves about 20% accuracy.

Table 1 shows the experimental results. Note that there are two variants which consistently get lower performance than their counterparts: using model II (versus using model I) and using committee decoding instead of Viterbi or MBR decoding. This suggests that the covariance matrices play a useful role during the learning process, but are not informative when performing decoding, since they are not used by Viterbi and MBR decoding. Interestingly, Smith and Eisner (2006) report a similar result for *structurally biased* DMV—a model that includes a parameter to control the length of the decoded dependencies. Their bias parameter is useful only during the learning process, but never during decoding. In general, the logistic normal distribution with model I outperforms substantially the baselines. It is interesting to note that LN-I outperforms Dirichlet-I and MLE even when using identity covariance matrices for initialization. The reason could be the fact that the logistic normal distribution, even when permitting only just diagonal covariance matrices (the case with identity covariance matrix initialization is weaker—we only initialize with diagonal matrices) allows to model the variance directly in the parameters. This is not possible with the Dirichlet distribution.

When we tested model II and committee decoding on other languages, the performance decrease was consistent. For the rest of the experiments, we report only MBR (and possibly Viterbi) decoding results using model I. The reason for the underperformance of model II could be the small number

language	tag set	training		development		test		baselines	
		tokens	sent.	tokens	sent.	tokens	sent.	A-R	A-L
English	34	55340	7179	35021	1700	49363	2416	30.2	20.4
Chinese	34	27357	4775	5824	350	7007	348	32.9	9.7
Portuguese	21	15976	2477	14558	907	5009	288	25.9	31.1
Turkish	29	18873	4497	7812	500	6288	623	68.2	4.3
Czech	47	67756	10674	32647	2535	33147	2535	24.4	28.3
Japanese	74	39121	10300	14666	1700	13648	1700	66.4	13.4

Table 2: Information about the data sets used in this paper. "Tag set" stands for the size of the part-of-speech tag set. Train, development and test columns show the number of tokens and number of sentences in each data set. The training set consists of sentences of length ten or less, as described in the text. The development set and the test set do not have any length restriction. The development set includes unannotated set of sentences from the respective language. A-R (A-L) stands for Attach-Right (Attach-Left), which are attachment accuracy baselines on the test set for all sentences. See text for details.

of parameters which is defined by the model. This small set of parameters cannot capture well the nuances across sentences in the data.

5.2 Additional Languages

Following Section 5.1, we experiment with other languages: Chinese, Portuguese, Turkish, Czech and Japanese.

- For Chinese, we used the Chinese treebank (Xue et al., 2004). We train on §1–270, use §301–1151 for development and test on §271–300.
- For Portuguese, we used the Bosque treebank (Afonso et al., 2002) from the CoNLL shared task in 2006 (Buchholz and Marsi, 2006).
- For Turkish, we used the METU-Sabancı treebank (Atalay et al., 2003; Oflazer et al., 2003) from the CoNLL shared task in 2006.
- For Czech, we used the Prague treebank (Hajič et al., 2000) from the CoNLL shared task in 2007 (Nivre et al., 2007).
- For Japanese, we used the VERBMOBIL Treebank for Japanese (Kawata and Bartels, 2000) from the CoNLL shared task in 2006.

Whenever using CoNLL shared task data, we used the first 80% of the data distributed in the shared task for training, and the rest was divided equally for development and testing. Table 2 gives statistics about the data sets used with the performance of the Attach-Right and Attach-Left baselines given for the whole test data. As in the case for English, sentences were stripped of words and punctuation, leaving part-of-speech tags for the unsupervised induction of dependency structure. All learning algorithms were run on sentences of length ten words or less. Note that strong performance is achieved for Turkish and Japanese by the Attach-Right baseline.

Results of running the different learning algorithms are given in Figure 4. Note that for Portuguese, the difference is much smaller between the EM baselines and logistic normal variational EM when only short sentences are considered, but there is a wider gap for longer sentences; the LN models appear to generalize better to longer sentences. For Turkish, no method outperforms Attach-Right, but there is still a big gap between variational EM with the logistic normal and the other EM baselines. The case is similar for Japanese, though logistic normal does outperform the Attach-Right baselines for shorter sentences. For Czech, it seems like Dirichlet and EM do somewhat better than the logistic normal prior, but performance of all four methods is close. It is conceivable that the approximation inherent in a projective syntax representation for the Czech sentences (whose gold-standard analyses have a relatively large fraction of nonprojective dependencies) interacts with different models in different ways.[6]

In general, the covariance matrices learned when initializing with the identity covariance matrix are rather sparse, but there is a high degree of variability across the diagonal (for the variance values learned). For the DMV, when using an identity initializer, diagonal matrices are the local optimum that is reached by the variational EM algorithm. When initializing the covariance matrices with the tag families initializer, the learned matrices are still rather sparse, but they have a larger number of significant correlations (for Portuguese, for example, using a t-test for testing the significance of the correlation, we found that 0.3% of the values in the covariance matrices had significant correlation).[7]

5.3 SLN with Nouns, Verbs, and Adjectives

We now turn to experiments where the partition structure lets parameters across multinomials covary, making use of the expressive power of the shared logistic normal distribution. We use a few simple heuristics to decide which partition structure $\mathcal{S}$ to use. Our heuristics rely mainly on the centrality of content words: nouns, verbs, and adjectives. For example, in the English treebank, the most common attachment errors (with the LN prior) happen with a noun (25.9%) or a verb (16.9%) parent. The fact that the most common errors happen with these attachments results from nouns and verbs being the most common parents in most of the data sets we experimented with.

Following this observation, we compare four different settings in our experiments (all SLN settings include one normal expert for each multinomial on its own, equivalent to the regular LN setting):

- TIEV: We add normal experts that tie all probabilities corresponding to a verbal parent (*any* verbal parent, using the coarse tags of Cohen et al., 2008). Let V be the set of part-of-speech tags that belong to the verb category. For each direction D (left or right), the set of multinomials of the form $\theta_c(\cdot \mid v, D)$, for $v \in V$, all share a normal expert. For each direction D and each boolean value B of the predicate $\text{first}_\mathbf{y}(\cdot)$, the set of multinomials $\theta_s(\cdot \mid v, D, B)$ for $v \in V$ share a normal expert.
- TIEN: This is the same as TIEV, only for nominal parents.

6. We note that we also experimented with other languages, including Hebrew and Arabic. We do not include these results, because in these cases all methods, including MLE, Dirichlet-I and LN-I performed badly (though Dirichlet-I and MLE could do better than LN-I). We believe that for these languages, the DMV is probably not the appropriate model. Developing better grammatical models for these languages is beyond the scope of this paper.
7. However, it is interesting to note that most of the elements of the covariance matrices were not exactly zero. For example, 90% of the values in the covariance matrices were larger (in absolute value) than 2.3×10^{-6}.

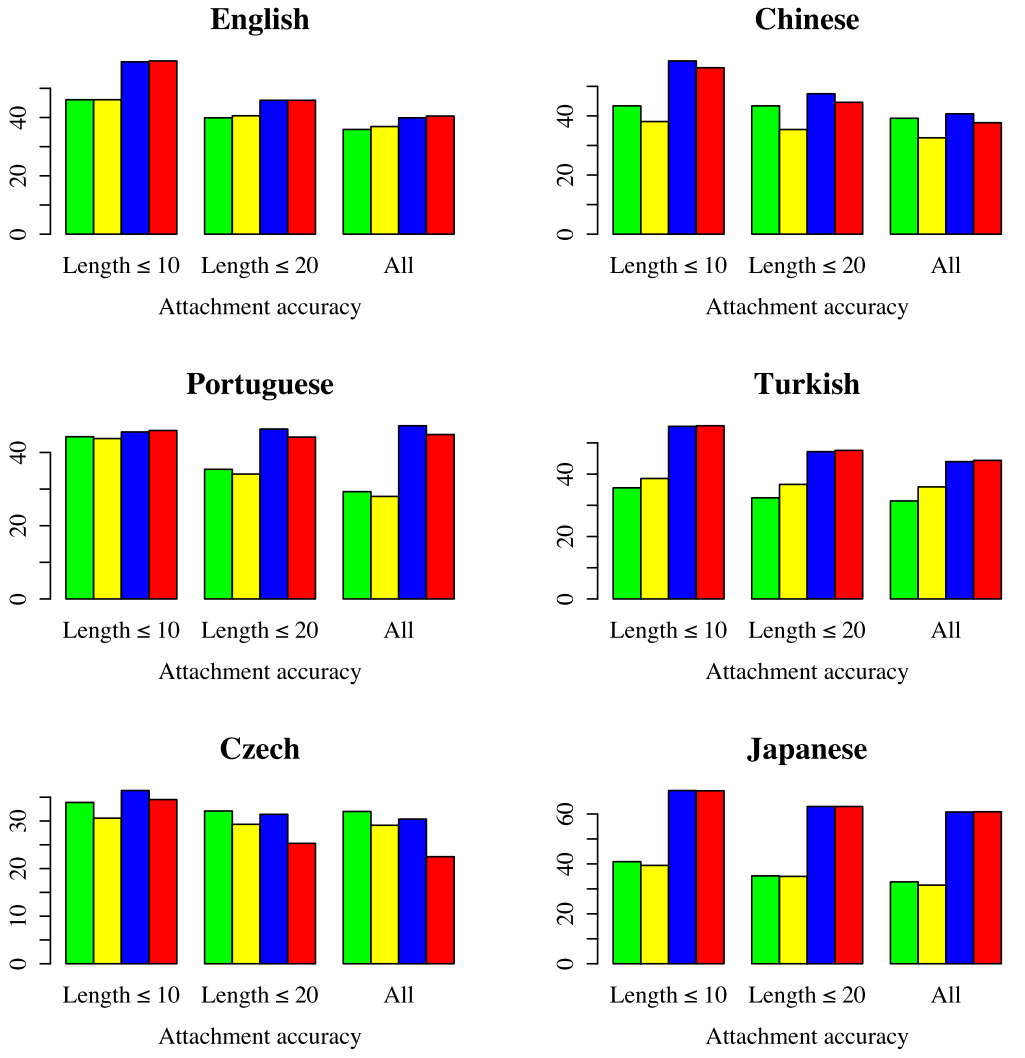


Figure 4: Attachment accuracy results for English (equivalent to Table 1), Chinese, Portuguese, Turkish, Czech and Japanese. The decoding mechanism used is MBR. Legend for the baselines: MLE (green, first column in each block); Dirichlet-I (yellow, second column); Legend for the methods in this paper: LN-I, $\Sigma_k^{(0)} = \mathbf{I}$ (blue, third column), and LN-I, families initializer (red, fourth column).

- TIEV&N: Tie both verbs and nouns (in separate partitions). This is equivalent to taking the union of the partition structures of the above two settings.

- TIEA: This is the same as TIEV, only for adjectival parents.

			English			Portuguese			Turkish		
			≤ 10	≤ 20	all	≤ 10	≤ 20	all	≤ 10	≤ 20	all
		MLE	46.1	39.9	35.9	44.3	35.4	29.3	35.6	32.4	31.4
		Dirichlet-I	46.1	40.6	36.9	43.8	34.1	28.0	38.6	36.7	35.9
		$\Sigma_k^{(0)} = \mathbf{I}$	59.1	45.9	40.5	45.6	**45.9**	**46.5**	55.3	47.2	44.0
		families	59.4	45.9	40.5	45.9	44.0	44.4	55.5	47.6	44.4
Trained with	English	TIEV	60.2	46.2	40.0	45.4	43.7	44.5	† 56.5	**48.7**	45.5
		TIEN	60.2	46.7	40.9	45.7	44.3	45.0	51.1	43.7	41.2
		TIEV&N	61.3	47.4	41.4	46.3	44.6	45.1	55.9	48.2	45.2
		TIEA	59.9	45.8	39.6	45.4	43.8	44.6	49.8	43.2	40.8
	Portuguese	TIEV	62.1	48.1	42.2	45.2	42.3	42.3	**56.7**	† 48.6	45.1
		TIEN	60.7	46.9	40.9	45.7	42.8	42.9	33.2	29.8	28.7
		TIEV&N	61.4	47.8	42.0	46.3	44.6	45.1	**56.7**	49.2	**46.0**
		TIEA	62.1	47.8	41.8	45.2	42.7	42.7	31.5	28.4	27.5
	Turkish	TIEV	**62.5**	**48.3**	**42.4**	45.4	43.2	43.7	55.2	47.3	44.0
		TIEN	61.0	47.2	41.2	45.9	43.9	44.4	45.1	39.8	37.8
		TIEV&N	† 62.3	**48.3**	† 42.3	**46.7**	44.3	44.6	55.7	**48.7**	45.5
		TIEA	† 62.3	48.0	42.1	45.1	43.2	43.7	38.6	34.0	32.5

Table 3: Attachment accuracy of different monolingual tying models and bilingual tying models in varying levels of difficulty imposed through a length filter (Sections 5.3 and 5.4). Monolingual results (Section 5.3) are described when the languages in both the column and the row are identical (blocks on the diagonal). Results for MLE and Dirichlet-I are identical to Figure 4. Results for $\Sigma_k^{(0)} = \mathbf{I}$ and families are identical to Table 1 and Figure 4. Each block contains the results of tying one language with the other, specifying performance for the column language. Results in bold denote best results in a column, and † marks figures that are not significantly worse (binomial sign test, $p < 0.05$).

Since learning a model with parameter tying can be computationally intensive, we first run the inference algorithm without parameter tying, and then add parameter tying to the rest of the inference algorithm's execution until convergence.

For the covariance matrices, we follow the setting described in Section 5.1. For each treebank, we divide the tags into twelve disjoint tag families. The covariance matrices for all dependency distributions were initialized with 1 on the diagonal, 0.5 between tags which belong to the same family, and 0 otherwise.

The results are given in the blocks on the diagonal of Table 3, where the languages in the columns and rows are identical. MBR decoding was used. For English, there are small improvements when adding the expressive power of SLN. The best results are achieved when tying both nouns and verbs together. Portuguese shows small benefits compared on shorter sentences, and when compared to the families-initialized LN-I model, but not the stronger identity-initialized LN-I model. For Turkish, tying across multinomials hurts performance.

5.4 Bilingual Experiments

Leveraging linguistic information from one language for the task of disambiguating another language has received considerable attention (Dagan, 1991; Yarowsky et al., 2001; Hwa et al., 2005; Smith and Smith, 2004; Snyder and Barzilay, 2008; Burkett and Klein, 2008). Usually such a setting requires a parallel corpus or other annotated data that ties between those two languages. One notable exception is Haghighi et al. (2008), where bilingual lexicons were learned from non-parallel monolingual corpora.

Our bilingual experiments use the data for English, Portuguese, and Turkish (two at a time), which are not parallel corpora, to train parsers for two languages at a time, jointly. Sharing information between two models is accomplished by softly tying grammar weights in the two hidden grammars.

For each pair of languages, we first merge the models for these two languages by taking a union of the multinomial families of each and the corresponding prior parameters. We then add a normal expert that ties between the parts of speech in the respective partition structures for both grammars together. Parts of speech are matched through the single coarse tag set. For example, with TIEV, let $V = V_{\text{Eng}} \cup V_{\text{Por}}$ be the set of part-of-speech tags which belong to the verb category for either the English or Portuguese treebank (to take an example). Then, we tie parameters for all part-of-speech tags in V. We tested this joint model for each of TIEV, TIEN, TIEV&N, and TIEA. After running the inference algorithm which learns the two models jointly, we use unseen data to test each learned model separately.

We repeat the generative story specifically for the bilingual setting, using the example of TIEV. For each language, there are normal experts for all part-of-speech tags, for the basic DMV. In addition, there are normal experts, for each language, that combine together all part-of-speech tags that belong to the verb category. Finally, there are normal experts, for the two languages *together*, that combine together all part-of-speech tags that belong to the verb category in either language. For each sentence in the corpus, the following two steps are conducted as before (model I): the normal experts are sampled from the SLN distribution and combined into multinomials to parameterize the DMV; a grammar derivation is sampled from the resulting DMV.

Table 3 presents the results for these experiments (blocks not on the diagonal). English grammar induction shows moderate gains when tied with Portuguese and strong gains with Turkish. Cohen and Smith (2009) reported qualitatively similar results when English was tied with Chinese. For Portuguese, there is not much gain from tying it with other languages, though it improves the performance of the other two languages. In general, the table shows that with the proper selection of pair of languages and multinomials to tie together, we can usually get improvement over the LN baselines and the technique is not harmful (cf. Turkish grammar induction with SLN, on its own). We note that selection of the multinomials to tie encodes prior knowledge about the languages. This knowledge simply requires being able to map fine-grained, treebank-specific part-of-speech tags to coarse categories. In addition, bilingual learning with SLN does not require bitext parsing at any point, which is an expensive operation. The runtime of the variational E-step for a sentence $\mathbf{x}$ is still cubic in the length of $\mathbf{x}$, as in EM, so that the runtime of the variational E-step scales in the multilingual case the same as it would be if we added an equivalent amount of data in the monolingual case.

Since the experiments reported here were conducted, others, notably Gillenwater et al. (2010) and Spitkovsky et al. (2010b), have reported performance surpassing ours, for some of the languages

in our experiments. Differences in the experimental settings prevent direct comparisons. Some of the improvements in dependency grammar induction are achieved because of techniques which are orthogonal to ours, such as improvements in the underlying grammar (instead of DMV; Headden et al., 2009; Gillenwater et al., 2010), and those techniques could be incorporated into the Bayesian model we described. Others are somewhat different (e.g., Viterbi training).

6. Discussion

We have shown that modeling covariance among grammar weights within a probabilistic grammar's multinomial distributions, *across* its distributions, and *across* grammars in two languages can have benefits for learning dependency structure in an unsupervised empirical Bayesian framework. This approach addresses one of the desiderata of Raiffa and Schaifer (1961) for prior distributions, "richness." The empirical benefits of modeling covariance, we have shown, are compelling.

We believe, however, that more remains to be done to incorporate prior linguistic knowledge into unsupervised grammar induction. Covariance structure is, perhaps, not the most *interpretable* kind of prior knowledge about grammars that might be brought to bear on learning. The empirical Bayesian paradigm explored here, and the use of variational approximations for coping with non-conjugacy, will be important tools in future work that brings together prior knowledge and unannotated data for grammar induction.

For natural language data, a direction for future work is to capture deeper linguistic phenomena. Here, background knowledge abounds: the entire field of theoretical linguistics has contributed both descriptive facts about the structure of specific natural languages and general theories about the way that structure is constrained. Viewing the logistic normal prior as local log-linear models (Section 3.4) is a first step towards encoding such prior knowledge. Similar to Berg-Kirkpatrick et al. (2010), it permits the use of arbitrary features in the parameterization of the grammar.

We note that our inference algorithm, described in detail in Appendix B, can be easily adapted to scenarios which do not necessarily use the multivariate normal distribution as the base distribution in the prior. The "softmax" can be applied to any multivariate sample to get a point in the probability simplex—perhaps capturing other tendencies in the data than covariance. The convenience of performing such an extension depends on the ability to effectively compute the moment generating function of the distribution replacing the multivariate Gaussian, in which case we can develop Equation 13 and proceed with optimizing the variational bound using this distribution.

7. Conclusion

In this paper we demonstrated the effectiveness of estimating probabilistic grammars in a Bayesian setting. We used the Bayesian setting to model covariance between the different parameters of probabilistic grammars. To model the covariance, we used the logistic normal distribution as a prior over the grammar parameters. In addition, we extended the logistic normal distribution to a new family of distributions, in order to model covariance across the multinomial family in a probabilistic grammar.

We proposed a variational inference algorithm for estimating the parameters of the probabilistic grammar, providing a fast, parallelizable,[8] and deterministic alternative to MCMC methods to approximate the posterior over derivations and grammar parameters.

8. We used a cluster running MapReduce (Dean and Ghemawat, 2004) to perform inference when training our models.

We experimented with grammar induction on six different languages, demonstrating the usefulness of our approach. Our experiments include a novel promising setting, in which syntactic trees are inferred in a bilingual setting that uses multilingual, non-parallel corpora. Notably, our approach tends to generalize better to longer sentences, despite learning (as in previous research) on short sentences. The focus of the experiments was on dependency grammar induction with the dependency model with valence. Our choice of the DMV is motivated by the fact that it is a widespread grammar for dependency grammar induction (Section 2.2), enabling us to tease apart the problem of estimation of the grammar from the problem of deciding on the grammar structure. Our inference algorithm, though, could be applied to any probabilistic grammar that has an efficient procedure, such as the inside-outside algorithm, for computing sufficient statistics in the form of expected counts of rule firing in grammar derivations.

Acknowledgments

The authors would like to thank the anonymous reviewers, Matthew Harrison, Mark Johnson, John Lafferty, and Eric Xing for their useful feedback and comments. The authors would also like to thank Yahoo! for making computational resources available. This research was supported by NSF grants IIS-0713265, IIS-0836431 and IIS-0915187.

Appendix A. Notation

Table 4 gives a table of notation for symbols used throughout this paper.

Appendix B. Variational Inference with Logistic Normal Priors

We give a derivation of a variational inference algorithm for model I, with the shared logistic normal distribution as a prior. The derivation is based on the one given in Blei and Lafferty (2006). The derivation for model II can be followed similarly, as explained below. For model I, we seek to find an approximation posterior function $q(\eta_1, ..., \eta_M, \mathbf{y}_1, ..., \mathbf{y}_M)$ that maximizes a lower bound (the negated variational free energy) on the log-likelihood, a bound which is achieved using Jensen's inequality (the following probability quantities should be understood as if we always condition on the grammar $\mathbf{G}$):

$$\begin{aligned}
&\sum_{m=1}^{M} \sum_{\mathbf{y}} \log p(\mathbf{x}_m, \mathbf{y} \mid \mu, \Sigma, \mathcal{S}) \\
&\geq \sum_{m=1}^{M} \left(\sum_{i=1}^{N} \mathbb{E}_q \left[\log p(\eta_{m,i} \mid \mu_i, \Sigma_i) \right] + \mathbb{E}_q \left[\log p(\mathbf{x}_m, \mathbf{y}_m \mid \eta_m, \mathcal{S}) \right] \right) + H(q). \qquad (10)
\end{aligned}$$

$H(\cdot)$ denotes the Shannon entropy.

We make a mean-field assumption, and assume that the posterior has the following form:

$$q(\eta_1, ..., \eta_M, \mathbf{y}_1, ..., \mathbf{y}_M) = \prod_{m=1}^{M} q_m(\eta_m, \mathbf{y}_m), \qquad (11)$$

	symbol	description
grammars and data	$\mathbf{G}$	grammar (for example, context-free grammar rules)
	M	number of observed sentences
	$\mathbf{x}_m$	mth observed sentence in the available data
	$\mathbf{y}_m$	inferred derivation grammatical structure for $\mathbf{x}_m$
	θ	parameters of a probabilistic grammar
	K	number of multinomials in the probabilistic grammar
	N_k	size of the kth multinomial of the probabilistic grammar
	$f_{k,i}(\mathbf{x},\mathbf{y})$	number of times the ith event fires in the kth multinomial in the derivations $\mathbf{x}$ and $\mathbf{y}$
priors	α	hyperparameters for the Dirichlet prior of a probabilistic grammar
	Σ	covariance matrices for the (shared) logistic normal prior of a probabilistic grammar
	μ	mean values for the (shared) logistic normal prior of a probabilistic grammar
	η	values drawn from the Gaussians for (S)LN, before the logistic transformation is applied
	$\mathcal{S}$	partition structure for the shared logistic normal distribution
	N	number of normal experts for the SLN
	l_n	length of nth normal expert (SLN)
	I_n	partition of the nth normal expert into segments mapping to multinomials in $\mathbf{G}$ (SLN)
	J_k	collection of segments of normal experts mapping to kth multinomial in $\mathbf{G}$ (SLN)
variational EM	$q_m(\theta,\mathbf{y})$	variational distribution which is used as an approximation posterior for the mth datum
	$\tilde{\mu}_{m,k,i}$	variational parameter for mean value of the ith event in the kth for the mth datum
	$\tilde{\sigma}_{m,k,i}$	variational parameter for variance of the ith event in the kth for the mth datum
	$\tilde{f}_{m,k,i}$	expected count of the ith event in the kth for the mth datum
	$\tilde{\psi}_{m,k,i}$	intermediate quantity aggregating variational parameters
	$\tilde{\zeta}_{m,k}$	variational parameter for the first-order Taylor approximation of LN's denominator

Table 4: Table of notation symbols used in this paper.

where

$$q_m(\eta_m,\mathbf{y}_m) = \left(\prod_{k=1}^{N}\prod_{i=1}^{L_k} q_m(\eta_{m,k,i} \mid \tilde{\mu}_{m,k,i},\tilde{\sigma}^2_{m,k,i})\right) \times q_m(\mathbf{y}_m),$$

and $q_m(\eta_{m,k,i} \mid \tilde{\mu}_{m,k,i},\tilde{\sigma}^2_{m,k,i})$ is a Gaussian with mean $\tilde{\mu}_{m,k,i}$ and variance $\tilde{\sigma}^2_{m,k,i}$. Note that this means that the *variational* distributions have a diagonal matrix for their covariance structure. The model covariance matrices (the hyperparameters Σ) can still have covariance structure. This selection of variational distributions makes inference much easier. The factorized form of Equation 11 implies the following identities:

$$\mathbb{E}_q\left[\log p(\eta_{m,i} \mid \mu_i, \Sigma_i)\right] = \mathbb{E}_{q_m}\left[\log p(\eta_{m,i} \mid \mu_i, \Sigma_i)\right],$$
$$\mathbb{E}_q\left[\log p(\mathbf{x}_m, \mathbf{y}_m \mid \eta_m, \mathcal{S})\right] = \mathbb{E}_{q_m}\left[\log p(\mathbf{x}_m, \mathbf{y}_m \mid \eta_m, \mathcal{S})\right],$$
$$H(q) = \sum_{m=1}^{N} H(q_m).$$

Let $\eta^C_{k,i}$ be an intermediate variable, denoting the average of the normal experts which appear in the partition structure and determine the value of the ith event in the kth multinomial of the grammar. More formally, we define the vector η^C_k of length N_k to be:

$$\eta^C_k \triangleq \frac{1}{|J_k|} \sum_{I_{r,j} \in J_k} \eta_{r,I_{r,j}}.$$

Unfolding the expectation with respect to $q_m(\mathbf{y}_m)$ in the second term in Equation 10, while recalling that θ_m is a deterministic function of η_m that averages different subvectors from the collection of multinomials η_m according to the partition structure $\mathcal{S}$, we have that:

$$\begin{aligned}
&\mathbb{E}_{q_m}\left[\log p(\mathbf{x}_m, \mathbf{y}_m \mid \eta_m, \mathcal{S})\right] \\
&= \mathbb{E}_{q_m(\eta_m)}\left[\sum_{k=1}^{K} \sum_{i=1}^{N_k} \underbrace{\sum_{\mathbf{y}} q_m(\mathbf{y}_m) f_{k,i}(\mathbf{x}_m, \mathbf{y}_m)}_{\tilde{f}_{m,k,i}} \log \theta_{m,k,i}\right] \\
&= \mathbb{E}_{q_m(\eta_m)}\left[\sum_{k=1}^{K} \sum_{i=1}^{N_k} \tilde{f}_{m,k,i}\left(\eta^C_{m,k,i} - \log \sum_{i'=1}^{N_k} \exp \eta^C_{m,k,i'}\right)\right],
\end{aligned} \tag{12}$$

where $\tilde{f}_{m,k,i}$ is the expected number of occurrences of the ith event in distribution k, under $q_m(\mathbf{y}_m)$. With many kinds of probabilistic grammars, this quantity can be computed using a dynamic programming algorithm like the forward-backward or inside-outside algorithm.

The logarithm term in Equation 12 is problematic because of the expectation with respect to $q_m(\eta_m)$. We approximate it with a first-order Taylor expansion, introducing $M \times K$ more variational parameters $\tilde{\zeta}_{m,k}$ for $m \in \{1, ..., M\}$ and $K \in \{1, ..., K\}$:

$$\log\left(\sum_{i'=1}^{N_k} \exp \eta^C_{m,k,i'}\right) \leq \log \tilde{\zeta}_{m,k} - 1 + \frac{1}{\tilde{\zeta}_{m,k}} \sum_{i'=1}^{N_k} \exp \eta^C_{m,k,i'}. \tag{13}$$

We note that the value $\mathbb{E}_{q_m(\eta_m)}\left[\exp(\eta^C_{m,k,i'})\right]$ can be calculated by evaluating the moment-generating function of the normal distribution $g(t) = \mathbb{E}_{q_m(\eta_m)}\left[\exp(t\eta^C_{m,k,i'})\right]$ at $t = 1$. We now have:

Algorithm 1: Variational EM for probabilistic grammars with LN prior

Input: initial parameters $\mu^{(0)}, \Sigma^{(0)}$, training data $\mathbf{x}$, and development data $\mathbf{x}'$
Output: learned parameters μ, Σ
$t \leftarrow 1$;
repeat
  Call E-Step for each training example $m = 1, ..., M$ *(Algorithm 2)*
  Call M-Step (Algorithm 3)
  $t \leftarrow t + 1$;
until *likelihood of held-out data,* $p(\mathbf{x}' \mid E[\mu^{(t)}])$*, decreases* ;
return $\mu^{(t)}, \Sigma^{(t)}$

$$
\begin{aligned}
&\mathbb{E}_{q_m}[\log p(\mathbf{x}_m, \mathbf{y}_m \mid \eta_m, \mathcal{S})] \\
&\geq \mathbb{E}_{q_m(\eta_m)}\left[\sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{m,k,i}\left(\eta_{m,k,i} - \log\tilde{\xi}_{m,k} + 1 - \frac{1}{\tilde{\xi}_{m,k}}\sum_{i'=1}^{N_k}\exp\eta_{m,k,i'}\right)\right] \\
&= \sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{m,k,i}\underbrace{\left(\tilde{\mu}_{m,k,i} - \log\tilde{\xi}_{m,k} + 1 - \frac{1}{\tilde{\xi}_{m,k}}\sum_{i'=1}^{N_k}\exp\left(\tilde{\mu}^{C}_{m,k,i} + \frac{(\tilde{\sigma}^{C}_{m,k,i})^2}{2}\right)\right)}_{\tilde{\psi}_{m,k,i}} \\
&= \sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{m,k,i}\tilde{\psi}_{m,k,i}
\end{aligned}
$$

where we use again the properties of the shared logistic normal distribution and rely on the partition structure $\mathcal{S}$ to define:

$$
\begin{aligned}
\tilde{\mu}^{C}_{m,k} &\triangleq \frac{1}{|J_k|}\sum_{I_{r,j}\in J_k}\tilde{\mu}_{m,r,I_{r,j}}, \\
(\tilde{\sigma}^{C}_{m,k})^2 &\triangleq \frac{1}{|J_k|^2}\sum_{I_{r,j}\in J_k}\tilde{\sigma}^{2}_{m,r,I_{r,j}}.
\end{aligned}
$$

Note the shorthand $\tilde{\psi}_{k,i}$ to denote an expression involving $\tilde{\mu}^{C}$, $\tilde{\sigma}^{C}$, and $\tilde{\xi}$.

The final form of our bound is:[9]

$$
\log p(\mathbf{x}, \mathbf{y} \mid \mu, \Sigma) \geq \left(\sum_{k=1}^{K}\mathbb{E}_q\left[\log p(\eta_k \mid \mu_k, \Sigma_k)\right]\right) + \left(\sum_{k=1}^{K}\sum_{i=1}^{N_k}\tilde{f}_{k,i}\tilde{\psi}_{k,i}\right) + H(q). \tag{14}
$$

Using an EM-style algorithm, we will alternate between finding the maximizing $q(\eta)$ and the maximizing $q(\mathbf{y})$. Maximization with respect to $q_m(\eta_m)$ is not hard, because $q(\eta)$ is parameterized. The following lemma shows that fortunately, finding the maximizing $q_m(\mathbf{y}_m)$, which we did not parameterize originally, is not hard either:

9. A tighter bound, based on a second-order approximation, was proposed in Ahmed and Xing (2007). We use a first-order approximation for simplicity, similar to Blei and Lafferty (2006).

Algorithm 2: E-Step (subroutine for Algorithm 1)

repeat

optimize for $\tilde{\mu}_{m,k}^{(t)}, k = 1, ..., K$: use conjugate gradient descent with

$$\frac{\partial B}{\partial \tilde{\mu}_{m,k,i}} = -\left((\Sigma_k^{(t-1)})^{-1})(\mu_k^{(t-1)} - \tilde{\mu}_{m,k})\right)_i - \tilde{f}_{m,k,i} + \sum_{i'=1}^{N_k} \left(\tilde{f}_{m,k,i'} / \tilde{\xi}_{m,k}\right) \exp\left(\frac{\tilde{\mu}_{k,i'} + \tilde{\sigma}_{k,i'}^2}{2}\right)$$

optimize $\tilde{\sigma}_{m,k}^{(t)}, k = 1, ..., K$: use Newton's method for each coordinate (with $\tilde{\sigma}_{m,k,i} > 0$) with

$$\frac{\partial B}{\partial \tilde{\sigma}_{m,k,i}^2} = -\frac{\Sigma_{k,ii}^{(t-1)}}{2} - \frac{\left(\sum_{i'=1}^{N_k} \tilde{f}_{m,k,i'}\right) \exp\left(\frac{\tilde{\mu}_{m,k,i} + \tilde{\sigma}_{m,k,i}^2}{2}\right)}{2\tilde{\xi}_{m,k}} + \frac{1}{2\tilde{\sigma}_{m,k,i}^2}$$

update $\tilde{\xi}_{m,k}^{(t)}, \forall k$:

$$\tilde{\xi}_{m,k}^{(t)} \leftarrow \sum_{i=1}^{N_k} \exp\left(\tilde{\mu}_{m,k,i}^{(t)} + \frac{(\tilde{\sigma}_{m,k,i}^{(t)})^2}{2}\right)$$

update $\tilde{\psi}_{m,k}^{(t)}, \forall k$:

$$\tilde{\psi}_{m,k,i}^{(t)} \leftarrow \tilde{\mu}_{m,k,i}^{(t)} - \log \tilde{\xi}_{m,k}^{(t)} + 1 - \frac{1}{\tilde{\xi}_{m,k}^{(t)}} \sum_{i'=1}^{N_k} \exp\left(\tilde{\mu}_{m,k,i}^{(t)} + \frac{(\tilde{\sigma}_{m,k,i}^{(t)})^2}{2}\right)$$

compute expected counts $\tilde{\mathbf{f}}_{m,k}^{(t)}, k = 1, ..., K$: use an inside-outside algorithm to re-estimate expected counts $\tilde{f}_{m,k,i}^{(t)}$ in weighted grammar $q(y)$ with weights $e^{\tilde{\psi}_m}$;

until *B does not change* ;

Lemma 2 *Let $r(\mathbf{y}_m \mid \mathbf{x}_m, e^{\tilde{\psi}_m})$ denote the conditional distribution over $\mathbf{y}_m$ given $\mathbf{x}_m$ defined as:*

$$r_m(\mathbf{y}_m \mid \mathbf{x}_m, e^{\tilde{\psi}}) = \frac{1}{Z_m(\tilde{\psi}_m)} \prod_{k=1}^{K} \prod_{i=1}^{N_k} \exp\left(\tilde{\psi}_{m,k,i} f_{m,k,i}(\mathbf{x}_m, \mathbf{y}_m)\right)$$

where $Z_m(\tilde{\psi}_m)$ is a normalization constant. Then $q_m(\mathbf{y}_m) = r_m(\mathbf{y}_m \mid \mathbf{x}_m, e^{\tilde{\psi}_m})$ maximizes the bound in Equation 14.

Proof First note that $H(q_m) = H(q_m(\eta_m \mid \tilde{\mu}_m, \tilde{\sigma}_m)) + H(q_m(\mathbf{y}_m))$. This means that the terms we are interested in maximizing from Equation 14 are the following, after plugging in $\tilde{f}_{m,k,i}$ explicitly:

$$L = \operatorname*{argmax}_{q_m(\mathbf{y}_m)} \sum_{\mathbf{y}_m} q_m(\mathbf{y}_m) \left(\sum_{k=1}^{K} \sum_{i=1}^{N_k} f_{m,k,i}(\mathbf{x}_m, \mathbf{y}_m) \tilde{\psi}_{m,k,i} \right) + H(q_m(\mathbf{y}_m)).$$

Algorithm 3: M-Step (subroutine for Algorithm 1)

Estimate $\mu^{(t)}$ and $\Sigma^{(t)}$ using the following maximum likelihood closed-form solution:

$$\mu_{k,i}^{(t)} \leftarrow \frac{1}{M}\sum_{m=1}^{M}\tilde{\mu}_{m,k,i}^{(t)}$$

$$\left[\Sigma_k^{(t)}\right]_{i,j} \leftarrow \frac{1}{M}\left(\sum_{m=1}^{M}\tilde{\mu}_{m,k,i}^{(t)}\tilde{\mu}_{m,k,j}^{(t)} + (\tilde{\sigma}^{(t)})_{m,k,i}^{2}\delta_{i,j} + M\mu_{k,i}^{(t)}\mu_{k,j}^{(t)} - \mu_{k,j}^{(t)}\sum_{m=1}^{M}\tilde{\mu}_{m,k,i}^{(t)} - \mu_{k,i}^{(t)}\sum_{m=1}^{M}\tilde{\mu}_{m,k,j}^{(t)}\right),$$

where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.

Then, note that:

$$L = \operatorname*{argmin}_{q_m(\mathbf{y}_m)} D_{\mathrm{KL}}\left(q_m(\mathbf{y}_m) \;\middle\|\; r_m(\mathbf{y}_m \mid \mathbf{x}_m, e^{\tilde{\Psi}_m})\right), \tag{15}$$

where D_{KL} denotes the KL divergence. To see that, combine the definition of KL divergence with the fact that $\sum_{k=1}^{K}\sum_{i=1}^{N_k} f_{m,k,i}(\mathbf{x},\mathbf{y})\tilde{\Psi}_{m,k,i} - \log Z_m(\tilde{\Psi}_m) = \log r_m(\mathbf{y}_m \mid \mathbf{x}_m, e^{\tilde{\Psi}_m})$ where $\log Z_m(\tilde{\Psi})$ does not depend on $q_m(\mathbf{y}_m)$. Equation 15 is minimized when $q_m = r_m$. ■

The above lemma demonstrates that the minimizing $q_m(\mathbf{y}_m)$ has the same form as the probabilistic grammar $\mathbf{G}$, only without having sum-to-one constraints on the weights (leading to the required normalization constant $Z_m(\tilde{\Psi}_m)$). As in classic EM with probabilistic grammars, we never need to represent $q_m(\mathbf{y}_m)$ explicitly; we need only $\tilde{\mathbf{f}}_m$, which can be calculated as expected feature values under $r_m(\mathbf{y}_m \mid \mathbf{x}_m, e^{\tilde{\Psi}_m})$ using dynamic programming.

Variational inference for model II is done similarly to model I. The main difference is that instead of having variational parameters for each $q_m(\eta_m)$, we have a single distribution $q(\eta)$, and the sufficient statistics from the inside-outside algorithm are used altogether to update it during variational inference.

Appendix C. Variational EM for Logistic-Normal Probabilistic Grammars

The algorithm for variational inference with probabilistic grammars using logistic normal prior is defined in Algorithms 1–3.[10] Since the updates for $\tilde{\zeta}_k^{(t)}$ are fast, we perform them after each optimization routine in the E-step (suppressed for clarity). There are variational parameters for each training example, indexed by m. We denote by B the variational bound in Equation 14. Our stopping criterion relies on the likelihood of a held-out set (Section 5) using a point estimate of the model.

10. An implementation of the algorithm is available at `http://www.ark.cs.cmu.edu/DAGEEM`. For simplicity, we give the vanilla logistic normal version of the algorithm in this appendix. The full version requires a more careful indexing and can be derived using the equations from Appendix B.

References

S. Afonso, E. Bick, R. Haber, and D. Santos. Floresta sinta(c)tica: a treebank for Portuguese. In *Proceedings of LREC*, 2002.

A. Ahmed and E. Xing. On tight approximate inference of the logistic normal topic admixture model. In *Proceedings of AISTATS*, 2007.

J. Aitchison. *The Statistical Analysis of Compositional Data*. Chapman and Hall, London, 1986.

H. Alshawi and A. L. Buchsbaum. Head automata and bilingual tiling: Translation with minimal representations. In *Proceedings of ACL*, 1996.

D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, pages 87–106, 1988.

N. B. Atalay, K. Oflazer, and B. Say. The annotation process in the Turkish treebank. In *Proceedings of LINC*, 2003.

J. Baker. Trainable grammars for speech recognition. In *The 97th meeting of the Acoustical Society of America*, 1979.

A. Banerjee. On Bayesian bounds. In *Proceedings of ICML*, 2006.

T. Berg-Kirkpatrick and D. Klein. Phylogenetic grammar induction. In *Proceedings of ACL*, 2010.

T. Berg-Kirkpatrick, A. Bouchard-Cote, J. DeNero, and D. Klein. Unsupervised learning with features. In *Proceedings of NAACL*, 2010.

D. M. Blei and J. D. Lafferty. Correlated topic models. In *Proceedings of NIPS*, 2006.

D. M. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

J. L. Boyd-Graber and D. M. Blei. Syntactic topic models. *CoRR*, abs/1002.4665, 2010.

P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based *n*-gram models of natural language. *Computational Linguistics*, 1990.

S. Buchholz and E. Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, 2006.

D. Burkett and D. Klein. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*, 2008.

G. Carroll and E. Charniak. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University, 1992.

E. Charniak and M. Johnson. Coarse-to-fine *n*-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, 2005.

S. F. Chen. Bayesian grammar induction for language modeling. In *Proceedings of ACL*, 1995.

D. Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, 2005.

A. Clark and F. Thollard. PAC-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5:473–497, 2004.

A. Clark, R. Eyraud, and A. Habrard. A polynomial algorithm for the inference of context free languages. In *Proceedings of ICGI*, 2008.

S. B. Cohen and N. A. Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of NAACL-HLT*, 2009.

S. B. Cohen and N. A. Smith. Empirical risk minimization with approximations of probabilistic grammars. In *NIPS*, 2010.

S. B. Cohen, K. Gimpel, and N. A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In *NIPS*, 2008.

S. B. Cohen, D. M. Blei, and N. A. Smith. Variational inference for adaptor grammars. In *Proceedings of NAACL*, 2010.

T. Cohn, S. Goldwater, and P. Blunsom. Inducing compact but accurate tree-substitution grammars. In *Proceedings of HLT-NAACL*, 2009.

M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, U. Penn., 1999.

M. Collins. Head-driven statistical models for natural language processing. *Computational Linguistics*, 29:589–637, 2003.

I. Dagan. Two languages are more informative than one. In *Proceedings of ACL*, 1991.

D. Das, N. Schneider, D. Chen, and N. A. Smith. Probabilistic frame-semantic parsing. In *Proceedings of ACL*, 2010.

C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38:1332–1348, 2005.

J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proceedings of OSDI*, 2004.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

Y. Ding and M. Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL*, 2005.

J. Eisner. Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of IWPT*, 1997.

J. R. Finkel, T. Grenager, and C. D. Manning. The infinite tree. In *Proceedings of ACL*, 2007.

H. Gaifman. Dependency systems and phrase-structure systems. *Information and Control*, 8, 1965.

K. Ganchev, J. Gillenwater, and B. Taskar. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL*, 2009.

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, and B. Taskar. Sparsity in dependency grammar induction. In *Proceedings of ACL*, 2010.

K. Gimpel and N. A. Smith. Feature-rich translation by quasi-synchronous lattice parsing. In *Proceedings of EMNLP*, 2009.

S. Goldwater. *Nonparametric Bayesian models of lexical acquisition*. PhD thesis, Brown University, 2006.

S. Goldwater and T. L. Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of ACL*, 2007.

A. Haghighi, P. Liang, T. Berg-Kirkpatrick, and D. Klein. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*, 2008.

J. Hajič, A. Böhmová, E. Hajičová, and B. Vidová Hldaká. The Prague dependency treebank: A three-level annotation scenario. *Treebanks: Building and Using Parsed Corpora*, pages 103–127, 2000.

W. P. Headden, M. Johnson, and D. McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of NAACL-HLT*, 2009.

G. E. Hinton. Products of experts. In *Proceedings of ICANN*, 1999.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*, 11(3):311–25, 2005.

R. Johansson and P. Nugues. LTH: Semantic structure extraction using nonprojective dependency trees. In *Proceedings of SemEval*, 2007.

M. Johnson. Why doesn't EM find good HMM POS-taggers? In *Proceedings EMNLP-CoNLL*, 2007.

M. Johnson, T. L. Griffiths, and S. Goldwater. Adaptor grammars: A framework for specifying compositional nonparameteric Bayesian models. In *NIPS*, 2006.

M. Johnson, T. L. Griffiths, and S. Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of NAACL*, 2007.

Y. Kawata and J. Bartels. Stylebook for the Japanese treebank in VERBMOBIL. Technical Report Verbmobil-Report 240, Seminar für Sprachwissenschaft, Univerisität Tübingen, 2000.

D. Klein and C. D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of ACL*, 2002.

D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430, 2003.

D. Klein and C. D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*, 2004.

K. Kurihara and T. Sato. Variational Bayesian grammar induction for natural language. In *Proceedings of ICGI*, 2006.

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

P. Liang, S. Petrov, M. Jordan, and D. Klein. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of EMNLP*, 2007.

D. Lin. A path-based transfer model for machine translation. In *Proceedings of COLING*, 2004.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330, 1993.

D. McAllester. PAC-Bayesian model averaging. *Machine Learning Journal*, 5:5–21, 2003.

D. Mimno, H. Wallach, and A. McCallum. Gibbs sampling for logistic normal topic models with graph-based priors. In *In Proceedings of NIPS Workshop on Analyzing Graphs*, 2008.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task, EMNLP-CoNLL*, 2007.

K. Oflazer, B. Say, D. Z. Hakkani-Tür, and G. Tür. Building a Turkish treebank. In A. Abeille, editor, *Building and Exploiting Syntactically-Annotated Corpora*. Kluwer, 2003.

F. C. N. Pereira and Y. Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of ACL*, 1992.

H. Raiffa and R. Schaifer. *Applied Statistical Decision Theory*. Wiley-Interscience, 1961.

M. Seeger. PAC-Bayesian generalization bounds for Gaussian processes. *Journal of Machine Learning Research*, 3:233–269, 2002.

D. A. Smith and N. A. Smith. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceedings of EMNLP*, 2004.

N. A. Smith. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. PhD thesis, Johns Hopkins University, 2006.

N. A. Smith and J. Eisner. Guiding unsupervised grammar induction using contrastive estimation. In *Proceedings of IJCAI Workshop on Grammatical Inference Applications*, 2005.

N. A. Smith and J. Eisner. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of COLING-ACL*, 2006.

B. Snyder and R. Barzilay. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL*, 2008.

V. Spitkovsky, H. Alshawi, and D. Jurafsky. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Proceedings of NAACL*, 2010a.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. Viterbi training improves unsupervised dependency parsing. In *Proceedings of CoNLL*, 2010b.

V. I. Spitkovsky, D. Jurafsky, and H. Alshawi. Profiting from mark-up: Hyper-text annotations for guided parsing. In *Proceedings of ACL*, 2010c.

L. Tesnière. *Élément de Syntaxe Structurale*. Klincksieck, 1959.

K. Toutanova and M. Johnson. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of NIPS*, 2007.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

M. Wang, N. A. Smith, and T. Mitamura. What is the Jeopardy model? a quasi-synchronous grammar for question answering. In *Proceedings of EMNLP*, 2007.

D. Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, 1997.

N. Xue, F. Xia, F.-D. Chiou, and M. Palmer. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30, 2004.

D. Yarowsky, G. Ngai, and R. Wicentoswki. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT*, 2001.

Journal of Machine Learning Research 11 (2010) 3053-3096 Submitted 5/10; Revised 9/10; Published 11/10

Inducing Tree-Substitution Grammars

Trevor Cohn TCOHN@DCS.SHEF.AC.UK
Department of Computer Science
University of Sheffield
Sheffield S1 4DP, UK

Phil Blunsom PBLUNSOM@COMLAB.OX.AC.UK
Computing Laboratory
University of Oxford
Oxford OX1 3QD, UK

Sharon Goldwater SGWATER@INF.ED.AC.UK
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK

Editor: Dorota Glowacka

Abstract

Inducing a grammar from text has proven to be a notoriously challenging learning task despite decades of research. The primary reason for its difficulty is that in order to induce plausible grammars, the underlying model must be capable of representing the intricacies of language while also ensuring that it can be readily learned from data. The majority of existing work on grammar induction has favoured model simplicity (and thus learnability) over representational capacity by using context free grammars and first order dependency grammars, which are not sufficiently expressive to model many common linguistic constructions. We propose a novel compromise by inferring a probabilistic *tree substitution grammar*, a formalism which allows for arbitrarily large tree fragments and thereby better represent complex linguistic structures. To limit the model's complexity we employ a Bayesian non-parametric prior which biases the model towards a sparse grammar with shallow productions. We demonstrate the model's efficacy on supervised phrase-structure parsing, where we induce a latent segmentation of the training treebank, and on unsupervised dependency grammar induction. In both cases the model uncovers interesting latent linguistic structures while producing competitive results.

Keywords: grammar induction, tree substitution grammar, Bayesian non-parametrics, Pitman-Yor process, Chinese restaurant process

1. Introduction

Inducing a grammar from a corpus of strings is one of the central challenges of computational linguistics, as well as one of its most difficult. Statistical approaches circumvent the theoretical problems with learnability that arise with non-statistical grammar learning (Gold, 1967), and performance has improved considerably since the early statistical work of Merialdo (1994) and Carroll and Charniak (1992), but the problem remains largely unsolved. Perhaps due to the difficulty of this unsupervised grammar induction problem, a more recent line of work has focused on a somewhat easier problem, where the input consists of a treebank corpus, usually in phrase-structure format,

and the task is to induce a grammar from the treebank that yields better parsing performance than the basic maximum-likelihood probabilistic context free grammar (PCFG). Examples of work on this kind of grammar induction, which we will refer to as *grammar refinement* because the learned grammars can be viewed as refinements of the treebank PCFG, include the symbol-splitting approach of Petrov and Klein (2007) and the tree-substitution grammars of Data-Oriented Parsing (Bod et al., 2003; Bod, 2003). Although the grammars induced by these systems are latent, the resulting parsers are supervised in the sense that the input to the learning system consists of strings and parses, and the goal is to learn how to parse new strings. Consequently, these systems do not remove the necessity of hand-annotating a large corpus, but they can potentially reduce the amount of engineering effort required to develop a successful statistical parser for a new language or domain, by obviating the need for complex hand-engineering of features, independence assumptions, and backoff schemes.

The desire for automatic grammar refinement methods highlights one possible reason why unsupervised grammar induction is so difficult. Simple models of syntactic structure such as hidden Markov models (HMMs) or PCFGs make strong independence assumptions that fail to capture the true complexity of language, so these models tend to learn something other than the desired structure when used in an unsupervised way. On the other hand, more complex models with, for example, head-lexicalized rules have too many free parameters to be learned successfully from unannotated data without the use of sophisticated backoff schemes. Thus, finding the right balance between learnability and complexity is critical to developing a successful model of grammar induction. We posit that this balance can be achieved by using a rich grammatical formalism coupled with a nonparametric Bayesian prior to limit the model's complexity. In this way the model can learn sufficiently complex structure to model the data, but will only do so when there is enough evidence to support such complexity; otherwise it will generalise to simpler structures. The model can therefore learn plausible structures from either small or large training samples.

We present here a model for automatically learning a *Probabilistic Tree Substitution Grammar* (PTSG) from either a treebank or strings. A PTSG is an extension to the PCFG in which nonterminals can rewrite as entire tree fragments (*elementary trees*), not just immediate children (Joshi, 2003; Bod et al., 2003). These large fragments can be used to encode non-local context, such as argument frames, gender agreement and idioms. A fundamental problem with PTSGs is that they are difficult to estimate, even in the supervised (grammar refinement) scenario where treebanked data is available. This is because treebanks are typically not annotated with their TSG derivations—how to decompose a tree into elementary tree fragments—instead the derivation needs to be inferred.

Probably the best-known previous work on inducing PTSGs is within the framework of Data-Oriented Parsing (DOP; Bod et al., 2003), which, like our model, has been applied in both supervised and unsupervised settings (Bod, 2003; Prescher et al., 2004; Zollmann and Sima'an, 2005; Zuidema, 2007; Bod, 2006).[1] DOP seeks to use as TSG productions all subtrees of the training corpus, an approach which makes parameter estimation difficult and led to serious problems with early estimation methods (Johnson, 2002), namely inconsistency for DOP1 (Bod, 2003) and overfitting of the maximum likelihood estimate (Prescher et al., 2004). More recent work on DOP estimation has tackled these problems, drawing from estimation theory to solve the consistency problem (Prescher et al., 2004; Zollmann and Sima'an, 2005), or using a grammar brevity heuristic to avoid the degeneracy of the MLE (Zuidema, 2007). Our work differs from DOP in that we use an ex-

1. Tree adjoining grammar induction (Chiang and Bikel, 2002; Xia, 2002) tackles a similar learning problem in the supervised case.

plicit generative model of TSG and a Bayesian prior for regularisation. The prior is nonparametric, which allows the model to learn a grammar of the appropriate complexity for the size of the training data. A further difference is that instead of seeking to use all subtrees from the training data in the induced TSG, our prior explicitly biases against such behaviour, such that the model learns a relatively compact grammar.[2] A final minor difference is that, because our model is generative, it assigns non-zero probability to all possible subtrees, even those that were not observed in the training data. In practice, unobserved subtrees will have very small probabilities.

We apply our model to the two grammar induction problems discussed above:

- **Inducing a TSG from a treebank.** This regime is analogous to the case of supervised DOP, where we induce a PTSG from a corpus of parsed sentences, and use this PTSG to parse new sentences. We present results using two different inference methods, training on either a subset of WSJ or on the full treebank. We report performance of 84.7% when training on the full treebank, far better than the 64.2% for a PCFG parser. These gains in accuracy are obtained with a grammar that is somewhat larger than the PCFG grammar, but still much smaller than the DOP all-subtrees grammar.

- **Inducing a TSG from strings.** As in other recent unsupervised parsing work, we adopt a dependency grammar (Mel′čuk, 1988) framework for the unsupervised regime. We use the split-head construction (Eisner, 2000; Johnson, 2007) to map between dependency and phrase-structure grammars, and apply our model to strings of POS tags. We report performance of 65.9% on the standard $\text{WSJ}_{\leq 10}$ data set, which is statistically tied with the best reported result on the task and considerably better than the EM baseline which obtains 46.1%. When evaluated on test data with no restriction on sentence length—a more realistic setting—our approach significantly improves the state-of-the-art.

Our work displays some similarities to previous work on both the grammar refinement and unsupervised grammar induction problems, but also differs in a number of ways. Aside from DOP, which we have already discussed, most approaches to grammar refinement can be viewed as symbol-splitting. That is, they allow each nonterminal to be split into a number of subcategories. The most notable examples of the symbol-splitting approach include Petrov et al. (2006), who use a likelihood-based splitting and merging algorithm, and Liang et al. (2007) and Finkel et al. (2007), who develop nonparametric Bayesian models. In theory, any PTSG can be recast as a PCFG with a sufficiently large number of subcategories (one for each unique subtree), so the grammar space of our model is a subspace of the symbol-splitting grammars. However, the number of nonterminals required to recreate our PTSG grammars in a PCFG would be exorbitant. Consequently, our model should be better able to learn specific lexical patterns, such as full noun phrases and verbs with their subcategorisation frames, while theirs are better suited to learning subcategories with larger membership, such as the days of the week or count versus mass nouns. The approaches are largely orthogonal, and therefore we expect that a PTSG with nonterminal refinement could capture both types of concept in a single model, thereby improving performance over either approach alone.

For the unsupervised grammar induction problem we adopt the Dependency Model with Valency (DMV; Klein and Manning, 2004) framework that is currently dominant for grammar induction (Cohen et al., 2009; Cohen and Smith, 2009; Headden III et al., 2009; Cohen et al., 2010;

2. The prior favours compact grammars by assigning the majority of probability mass to few productions, and very little (but non-zero) mass to other productions. In practice we use Markov Chain Monte Carlo sampling for inference which results in sparse counts with structural zeros, thus permitting efficient representation.

Spitkovsky et al., 2010). The first grammar induction models to surpass a trivial baseline concentrated on the task of inducing unlabelled bracketings for strings and were evaluated against treebank bracketing gold standard (Clark, 2001; Klein and Manning, 2002). Subsequently the DMV model has proved more attractive to researchers, partly because it defined a well founded generative stochastic grammar, and partly due to the popularity of dependency trees in many natural language processing (NLP) tasks. Recent work on improving the original DMV model has focused on three avenues: smoothing the head-child distributions (Cohen et al., 2009; Cohen and Smith, 2009; Headden III et al., 2009), initialisation (Headden III et al., 2009; Spitkovsky et al., 2010), and extending the conditioning distributions (Headden III et al., 2009). Our work falls into the final category: by extending the DMV CFG model to a TSG we increase the conditioning context of head-child decisions within the model, allowing the grammar to directly represent groups of linked dependencies.

Adaptor Grammars (Johnson et al., 2007b) are another recent nonparametric Bayesian model for learning hierarchical structure from strings. They instantiate a more restricted class of tree-substitution grammar in which each subtree expands completely, with only terminal symbols as leaves. Since our model permits nonterminals as subtree leaves, it is more general than Adaptor Grammars. Adaptor Grammars have been applied successfully to induce labeled bracketings from strings in the domains of morphology and word segmentation (Johnson, 2008a,b; Johnson and Goldwater, 2009) and very recently for dependency grammar induction (Cohen et al., 2010). The latter work also introduced a variational inference algorithm for Adaptor Grammar inference; we use a sampling method here.

The most similar work to that presented here is our own previous work (Cohn et al., 2009; Cohn and Blunsom, 2010), in which we introduced a version of the model described here, along with two other papers that independently introduced similar models (Post and Gildea, 2009; O'Donnell et al., 2009). Cohn et al. (2009) and Post and Gildea (2009) both present models based on a Dirichlet process prior and provide results only for the problem of grammar refinement, whereas in this article we develop a newer version of our model using a Pitman-Yor process prior, and also show how it can be used for unsupervised learning. These extensions are also reported in our recent work on dependency grammar induction (Blunsom and Cohn, 2010), although in this paper we present a more thorough exposition of the model and experimental evaluation. O'Donnell et al. (2009) also use a Pitman-Yor process prior (although their model is slightly different from ours) and present unsupervised results, but their focus is on cognitive modeling rather than natural language processing, so their results are mostly qualitative and include no evaluation of parsing performance on standard corpora.

To sum up, although previous work has included some aspects of what we present here, this article contains several novel contributions. Firstly we present a single generative model capable of both supervised and unsupervised learning, to induce tree substitution grammars from either trees or strings. We demonstrate that in both settings the model outperforms maximum likelihood baselines while also achieving results competitive with the best current systems. The second main contribution is to provide a thorough empirical evaluation in both settings, examining the effect of various conditions including data size, sampling method and parsing algorithm, and providing an analysis of the structures that were induced.

In the remainder of this article, we briefly review PTSGs in Section 2 before presenting our model, including versions for both constituency and dependency parsing, in Section 3. In Section 4 we introduce two different Markov Chain Monte Carlo (MCMC) methods for inference: a local Gibbs sampler and a blocked Metropolis-Hastings sampler. The local sampler is much simpler but

is only applicable in the supervised setting, where the trees are observed, whereas the Metropolis-Hastings sampler can be used in both supervised and unsupervised settings and for parsing. We discuss how to use the trained model for parsing in Section 5, presenting three different parsing algorithms. Experimental results for supervised parsing are provided in Section 6, where we compare the different training and parsing methods. Unsupervised dependency grammar induction experiments are described in Section 7, and we conclude in Section 8.

2. Tree-substitution grammars

A *Tree Substitution Grammar*[3] (TSG) is a 4-tuple, $G = (T, N, S, R)$, where T is a set of *terminal symbols*, N is a set of *nonterminal symbols*, $S \in N$ is the distinguished *root nonterminal* and R is a set of productions (rules). The productions take the form of *elementary trees*—tree fragments[4] of height ≥ 1—where each internal node is labelled with a nonterminal and each leaf is labelled with either a terminal or a nonterminal. Nonterminal leaves are called *frontier nonterminals* and form the substitution (recursion) sites in the generative process of creating trees with the grammar. For example, in Figure 1b the S $\rightarrow$ NP (VP (V hates) NP) production rewrites the S nonterminal as the fragment (S NP (VP (V hates) NP)).[5] This production has the two NPs as its frontier nonterminals.

A *derivation* creates a tree by starting with the root symbol and rewriting (substituting) it with an elementary tree, then continuing to rewrite frontier nonterminals with elementary trees until there are no remaining frontier nonterminals. We can represent derivations as sequences of elementary trees $\mathbf{e}$, where each elementary tree is substituted for the left-most frontier nonterminal of the tree being generated. Unlike Context Free Grammars (CFGs) a syntax tree may not uniquely specify the derivation, as illustrated in Figure 1 which shows two derivations using different elementary trees to produce the same tree.

A *Probabilistic Tree Substitution Grammar* (PTSG), like a PCFG, assigns a probability to each rule in the grammar, denoted $P(e|c)$ where the elementary tree e rewrites nonterminal c. The probability of a derivation $\mathbf{e}$ is the product of the probabilities of its component rules. Thus if we assume that each rewrite is conditionally independent of all others given its root nonterminal c (as in standard TSG models),[6] then we have

$$P(\mathbf{e}) = \prod_{c \rightarrow e \in \mathbf{e}} P(e|c)\,. \tag{1}$$

The probability of a tree, t, and string of words, $\mathbf{w}$, are given by

$$P(t) = \sum_{\mathbf{e}:\text{tree}(\mathbf{e})=t} P(\mathbf{e}) \quad \text{and}$$

$$P(\mathbf{w}) = \sum_{t:\text{yield}(t)=\mathbf{w}} P(t)\,,$$

3. A TSG is a *Tree Adjoining Grammar* (TAG; Joshi, 2003) without the adjunction operator, which allows insertions at internal nodes in the tree. This operation allows TAGs to describe the set of mildly context sensitive languages. A TSG in contrast can only describe the set of context free languages.
4. Elementary trees of height 1 correspond to productions in a context free grammar.
5. We use bracketed notation to represent tree structures as linear strings. The parenthesis indicate the hierarchical structure, with the first argument denoting the node label and the following arguments denoting child trees. The nonterminals used in our examples denote nouns, verbs, etc., and their respective phrasal types, using a simplified version of the Penn treebank tag set (Marcus et al., 1993).
6. Note that this conditional independence does not hold for our model because (as we will see in Section 3) we integrate out the model parameters.

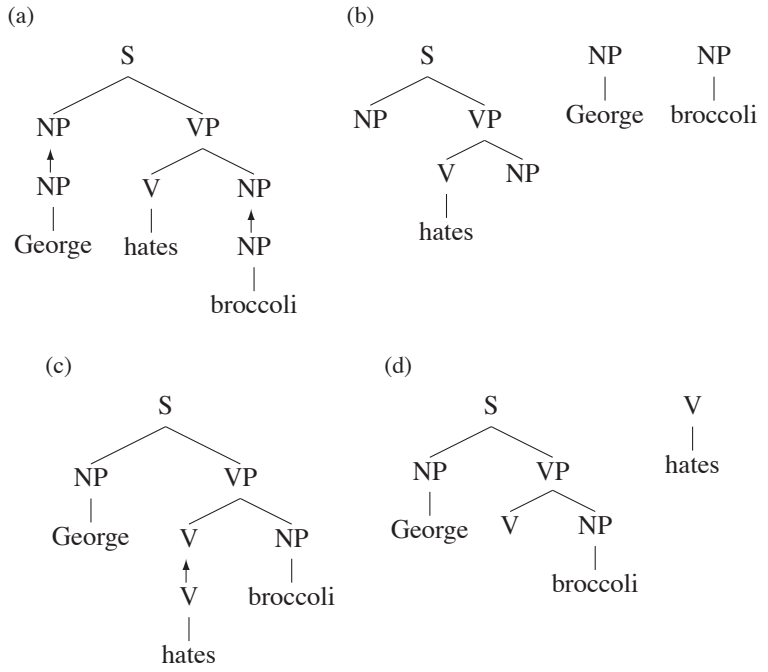


Figure 1: Example derivations for the same tree, where arrows indicate substitution sites. The left figures (a) and (c) show two different derivations and the right figures (b) and (d) show the elementary trees used in the respective derivation.

respectively, where tree($\mathbf{e}$) returns the tree for the derivation $\mathbf{e}$ and yield(t) returns the string of terminal symbols at the leaves of t.

Estimating a PTSG requires learning the sufficient statistics for $P(e|c)$ in (1) based on a training sample. Estimation has been done in previous work in a variety of ways, for example using heuristic frequency counts (Bod, 1993), a maximum likelihood estimate (Bod, 2000) and heldout estimation (Prescher et al., 2004). Parsing involves finding the most probable tree for a given string, that is, $\arg\max_t P(t|\mathbf{w})$. This is typically simplified to finding the most probable derivation, which can be done efficiently using the CYK algorithm. A number of improved algorithms for parsing have been reported, most notably a Monte-Carlo approach for finding the maximum probability tree (Bod, 1995) and a technique for maximising labelled recall using inside-outside inference in a PCFG reduction grammar (Goodman, 1998).

2.1 Dependency Grammars

Due to the wide availability of annotated treebanks, phrase structure grammars have become a popular formalism for building supervised parsers, and we will follow this tradition by using phrase structure trees from the Wall Street Journal corpus (Marcus et al., 1993) as the basis for our supervised grammar induction experiments (grammar refinement). However, the choice of formalism for unsupervised induction is a more nuanced one. The induction of phrase-structure grammars is notoriously difficult, since these grammars contain two kinds of ambiguity: the constituent structure and the constituent labels. In particular, constituent labels are highly ambiguous: firstly we don't know *a priori* how many there are, and secondly labels that appear high in a tree (e.g., an *S* category

Figure 2: An unlabelled dependency analysis for the example sentence *George hates broccoli*. The artificial ROOT node denotes the head of the sentence.

for a clause) rely on the correct inference of all the latent labels above and below them. Much of the recent work on unsupervised grammar induction has therefore taken a different approach, focusing on inducing dependency grammars (Mel′čuk, 1988). In applying our model to unsupervised grammar induction we follow this trend by inducing a dependency grammar. Dependency grammars represent the structure of language through directed links between words, which relate words (heads) with their syntactic dependents (arguments). An example dependency tree is shown in Figure 2, where directed arcs denote each word's arguments (e.g., hates has two arguments, 'George' and 'broccoli'). Dependency grammars are less ambiguous than phrase-structure grammars since the set of possible constituent labels (heads) is directly observed from the words in the sentence, leaving only the induction challenge of determining the tree structure. Most dependency grammar formalisms also include labels on the links between words, denoting, for example, subject, object, adjunct etc. In this work we focus on inducing unlabelled directed dependency links and assume that these links form a projective tree (there are no crossing links, which correspond to discontinuous constituents). We leave the problem of inducing labeled dependency grammars to further work.

Although we will be inducing dependency parses in our unsupervised experiments, we define our model in the following section using the formalism of a phrase-structure grammar. As detailed in Section 7, the model can be used for dependency grammar induction by using a specially designed phrase-structure grammar to represent dependency links.

3. Model

In defining our model, we focus on the unsupervised case, where we are given a corpus of text strings $\mathbf{w}$ and wish to learn a tree-substitution grammar G that we can use to infer the parses for our strings and to parse new data. (We will handle the supervised scenario, where we are given observed trees $\mathbf{t}$, in Section 4; we treat it as a special case of the unsupervised model using additional constraints during inference.) Rather than inferring a grammar directly, we go through an intermediate step of inferring a distribution over the derivations used to produce $\mathbf{w}$, that is, a distribution over sequences of elementary trees $\mathbf{e}$ that compose to form $\mathbf{w}$ as their yield. We will then essentially read the grammar off the elementary trees, as described in Section 5. Our problem therefore becomes one of identifying the posterior distribution of $\mathbf{e}$ given $\mathbf{w}$, which we can do using Bayes' Rule,

$$P(\mathbf{e}|\mathbf{w}) \propto P(\mathbf{w}|\mathbf{e})P(\mathbf{e})\,.$$

Note that any sequence of elementary trees uniquely specifies a corresponding sequence of words: those words that can be read off the leaves of the elementary trees in sequence. Therefore, given a sequence of elementary trees $\mathbf{e}$, $P(\mathbf{w}|\mathbf{e})$ either equals 1 (if $\mathbf{w}$ is consistent with $\mathbf{e}$) or 0 (otherwise). Thus, in our model, all the work is done by the prior distribution over elementary trees,

$$P(\mathbf{e}|\mathbf{w}) \propto P(\mathbf{e})\,\delta(w(\mathbf{e}),\mathbf{w})\,,$$

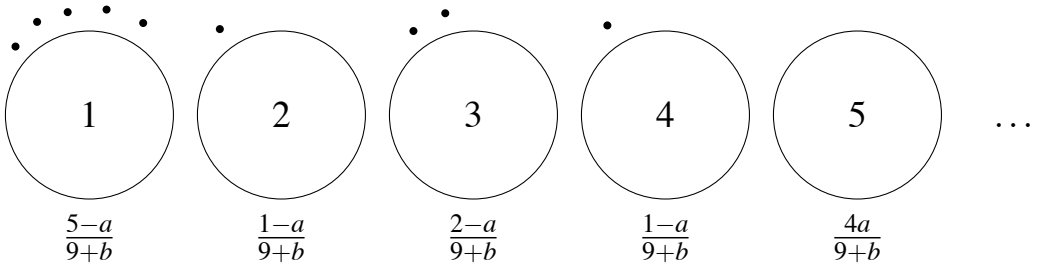


Figure 3: An example of the Pitman-Yor Chinese restaurant process with $\mathbf{z}_{-10} = (1,2,1,1,3,1,1,4,3)$. Black dots indicate the number of customers sitting at each table, and the value listed below table k is $P(z_{10} = k|\mathbf{z}_{-10})$.

where δ is the Kronecker delta and $w(\mathbf{e}) = \text{yield}(\text{tree}(\mathbf{e}))$ returns the string yield of the tree defined by the derivation $\mathbf{e}$.

Because we have no way to know ahead of time how many elementary trees might be needed to account for the data, we use a nonparametric Bayesian prior, specifically the Pitman-Yor process (PYP) (Pitman, 1995; Pitman and Yor, 1997; Ishwaran and James, 2003), which is a generalization of the more widely known Dirichlet process (Ferguson, 1973). Drawing a sample from a PYP (or DP) yields a probability distribution G with countably infinite support. The PYP has three parameters: a *discount parameter* a, a *strength parameter* b, and a *base distribution* P_{E}. Informally, the base distribution determines which items will be in the support of G (here, we will define P_{E} as a distribution over elementary trees, so that G is also a distribution over elementary trees), and the discount and strength parameters a and b determine the shape of G. The discount parameter a ranges from 0 to 1; when $a = 0$, the PYP reduces to a Dirichlet process, in which case the strength parameter b is known as the *concentration parameter* and is usually denoted with α. We discuss the roles of a and b further below.

Assuming an appropriate definition for P_{E} (we give a formal definition below), we can use the PYP to define a distribution over sequences of elementary trees $\mathbf{e} = e_1 \dots e_n$ as follows:

$$\begin{aligned} G|a,b,P_{\text{E}} &\sim \text{PYP}(a,b,P_{\text{E}}) \\ e_i|G \qquad &\sim G. \end{aligned} \tag{2}$$

In this formulation, G is an infinite distribution over elementary trees drawn from the PYP prior, and the e_i are drawn *iid* from G. However, since it is impossible to explicitly represent an infinite distribution, we integrate over possible values of G, which induces dependencies between the e_i. Perhaps the easiest way to understand the resulting distribution over $\mathbf{e}$ is through a variant of the Chinese restaurant process (CRP; Aldous, 1985; Pitman, 1995) that is often used to explain the Dirichlet process. Imagine a restaurant with an infinite number of tables, each with an infinite number of seats. Customers enter the restaurant one at a time and seat themselves at a table. If z_i is the index of the table chosen by the ith customer, then the Pitman-Yor Chinese Restaurant Process (PYCRP) defines the distribution

$$P(z_i = k|\mathbf{z}_{-i}) = \begin{cases} \frac{n_k^- - a}{i-1+b} & 1 \le k \le K^- \\ \frac{K^- a + b}{i-1+b} & k = K^- + 1 \end{cases},$$

where $\mathbf{z}_{-i}$ is the seating arrangement of the $i-1$ previous customers, n_k^- is the number of customers in $\mathbf{z}_{-i}$ who are seated at table k, $K^- = K(\mathbf{z}_{-i})$ is the total number of tables in $\mathbf{z}_{-i}$, and $z_1 = 1$ by definition. Figure 3 illustrates. When $a = 0$, this process reduces to the standard Chinese restaurant process. Like the CRP, the PYCRP is exchangeable and produces a power-law distribution on the number of customers at each table (Pitman, 2006). The hyperparameters a and b together control the manner of the clustering, although the difference between the two is rather subtle. A high value of b will bias towards more clusters irrespective of their individual sizes, only accounting for their aggregate size. In contrast a large $a \to 1$ will bias the clustering towards smaller individual clusters.

The PYCRP produces a sequence of integers $\mathbf{z}$ whose joint probability is

$$\begin{aligned} P(\mathbf{z}) &= \prod_{i=1}^{n} P(z_i|\mathbf{z}_{1\ldots i-1}) \\ &= 1 \cdot \prod_{i=2}^{n} P(z_i|\mathbf{z}_{1\ldots i-1}) \\ &= \left(\prod_{j=1}^{n-1} \frac{1}{j+b}\right)\left(\prod_{k=1}^{K(\mathbf{z})-1} (ka+b)\right)\left(\prod_{k=1}^{K(\mathbf{z})} \prod_{j=1}^{n_k^- - 1} (j-a)\right) \\ &= \frac{\Gamma(1+b)}{\Gamma(n+b)}\left(\prod_{k=1}^{K-1}(ka+b)\right)\left(\prod_{k=1}^{K} \frac{\Gamma(n_k^- - a)}{\Gamma(1-a)}\right), \end{aligned} \tag{3}$$

where K is the total number of tables in $\mathbf{z}$ and Γ is the gamma function. In order to produce a sequence of elementary trees $\mathbf{e}$ we need to introduce a second step in the process. We can do so by imagining that each table in the restaurant is labelled with an elementary tree, with $\ell(\mathbf{z}) = \ell_1 \ldots \ell_K$ being the trees labelling each table. Whenever a customer sits at a previously unoccupied table, a label is chosen for that table according to the base distribution P_{E}, and this label is used by all following customers at that table, as illustrated in Figure 4. We define e_i to be ℓ_{z_i}, the label of the table chosen by the ith customer. This yields the following conditional distribution on e_i:

$$\begin{aligned} P(e_i = e|\mathbf{z}_{-i}, \ell(\mathbf{z}_{-i})) &= \sum_{k=1}^{K(\mathbf{z}_{-i})} \delta(\ell_k, e)\frac{n_k^{(\mathbf{z}_{-i})} - a}{i-1+b} + \frac{K(\mathbf{z}_{-i})a+b}{i-1+b} P_{\text{E}}(e) \\ &= \frac{n_e^- - K_e(\mathbf{z}_{-i})a + (K(\mathbf{z}_{-i})a+b)P_{\text{E}}(e)}{i-1+b}, \end{aligned} \tag{4}$$

where K_e^- is the number of tables labelled with e in $\mathbf{z}_{-i}$, and δ is the Kronecker delta. The probability of an entire sequence of elementary trees is

$$P(\mathbf{e}) = \sum_{\mathbf{z},\ell} P(\mathbf{e}, \mathbf{z}, \ell),$$

where $P(\mathbf{e}, \mathbf{z}, \ell) = 0$ except when $\ell_{z_i} = e_i$ for all i, in which case

$$\begin{aligned} P(\mathbf{e}, \mathbf{z}, \ell) &= P(\mathbf{z}, \ell) = P(\mathbf{z})P(\ell|\mathbf{z}) \\ &= \frac{\Gamma(1+b)}{\Gamma(n+b)}\left(\prod_{k=1}^{K-1}(ka+b)\right)\left(\prod_{k=1}^{K} \frac{\Gamma(n_k^- - a)}{\Gamma(1-a)} P_{\text{E}}(\ell_k)\right), \end{aligned}$$

where K is the total number of tables in $\mathbf{z}$.

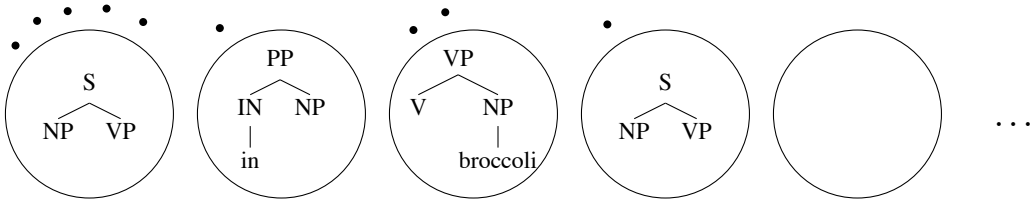


Figure 4: The Pitman-Yor process, illustrated as a labelled Chinese restaurant process. In this example, $\mathbf{z}_{-10} = (1,2,1,1,3,1,1,4,3)$ and each table k is labelled with an elementary tree ℓ_k. Black dots indicate the number of occurrences of each tree in $\mathbf{e} = (\ell_1,\ell_2,\ell_1,\ell_1,\ell_3,\ell_1,\ell_1,\ell_4,\ell_3)$. In this illustration, which corresponds to the model given in (2), a single Pitman-Yor process is used to generate all elementary trees, so the trees do not necessarily fit together properly. Our complete model, defined in (5), would have a separate Pitman-Yor restaurant for each root category.

Equation 4 shows that, like other PYP and DP models, this model can be viewed as a *cache model*, where e_i can be generated in one of two ways: by drawing from the base distribution or by drawing from a cache of previously generated elementary trees, where the probability of any particular elementary tree is proportional to the discounted frequency of that tree. This view makes it clear that the model embodies a "rich-get-richer" dynamic in which a few elementary trees will occur with high probability, but many will occur only once or twice, as is typical of natural language.

In the model just defined, a single PYP generates all of the elementary trees in $\mathbf{e}$. Notice, however, that these elementary trees might not tile together properly to create full syntax trees. For example, in Figure 4, e_1 = (S NP VP) and e_2 = (PP (IN in) NP), where the first substitution site in e_1 is an NP, but the root of e_2 is a PP, so e_2 cannot be used to expand e_1. To solve this problem, we modify the model so that there is a separate PYP for each non-terminal category c, with a base distribution conditioned on c. The distribution over elementary trees with root category c is defined as

$$\begin{aligned} G_c|a_c,b_c,P_{\text{E}} &\sim \text{PYP}(a_c,b_c,P_{\text{E}}(\cdot|c)) \\ e|c,G_c &\sim G_c\,, \end{aligned} \tag{5}$$

where $P_{\text{E}}(\cdot|c)$ is a distribution over the infinite space of elementary trees rooted with c, and a_c and b_c are the PYP hyper-parameters for non-terminal c. We elect not to tie together the values of these hyper-parameters as these control the tendency to infer larger or smaller sets of elementary trees from the observed data; we expect the distribution over productions to differ substantially between non-terminals. To generate $\mathbf{e}$, we now draw e_1 from G_S, giving us an elementary tree with frontier nodes $c_1 \ldots c_m$. We then draw $e_2 \ldots e_m$ in turn from $G_{c_1} \ldots G_{c_m}$. We continue in this fashion until a full tree is generated, at which point we can start again with a draw from G_S.

Integrating over G_c, we obtain the following distribution over e_i, now conditioned on its root category as well as the previously generated table labels and assignments:

$$P(e_i = e|c,\mathbf{z}_{-i},\ell(\mathbf{z}_{-i})) = \frac{n_e^- - K_e^- a_c + (K_c^- a_c + b_c)P_{\text{E}}(e|c)}{n_c^- + b_c}\,, \tag{6}$$

where $K_c^- = \sum_{e:\text{root}(e)=c} K_e^-$ is the total number of tables for nonterminal c, n_e^- is the number of times e has been used to rewrite c and $n_c^- = \sum_{e:\text{root}(e)=c} n_e^-$ is the total count of rules rewriting c. As

before, the $^-$ superscript denotes that the counts are calculated over the previous elementary trees, $\mathbf{e}_{-i}$, and their seating arrangements, $\mathbf{z}_{-i}$.

Finally, we turn to the definition of the base distribution over elementary trees, P_{E}. Recall that in an elementary tree, each internal node is labelled with a non-terminal category symbol and each frontier (leaf) node is labelled with either a non-terminal or a terminal symbol. Given a probabilistic context-free grammar R, we assume that elementary trees are generated (conditioned on the root non-terminal c) using the following generative process. First, choose a PCFG production $c \rightarrow \alpha$ for expanding c according to the distribution given by R. Next, for each non-terminal in α decide whether to stop expanding (creating a non-terminal frontier node, also known as a substitution site) or to continue expanding. If the choice is to continue expanding, a new PCFG production is chosen to expand the child, and the process continues recursively. The generative process completes when the frontier is composed entirely of substitution sites and terminal symbols.

Assuming a fixed distribution P_{C} over the rules in R, this generative process leads to the following distribution over elementary trees:

$$P_{\mathrm{E}}(e|c) = \prod_{i \in I(e)} (1 - s_{c_i}) \prod_{f \in F(e)} s_{c_f} \prod_{c' \rightarrow \alpha \in e} P_{\mathrm{C}}(\alpha|c') \,, \tag{7}$$

where $I(e)$ are the set of internal nodes in e excluding the root, $F(e)$ are the set of frontier non-terminal nodes, c_i is the non-terminal symbol for node i and s_c is the probability of stopping expanding a node labelled c. We treat s_c as a parameter which is estimated during training, as described in Section 4.3. In the supervised case it is reasonable to assume that P_{C} is fixed; we simply use the maximum-likelihood PCFG distribution estimated from the training corpus (i.e., $P_{\mathrm{C}}(\alpha|c')$ is simply the relative frequency of the rule $c' \rightarrow \alpha$). In the unsupervised case, we will infer P_{C}; this requires extending the model to assume that P_{C} is itself drawn from a PYP prior with a uniform base distribution. We describe this extension below, along with its associated changes to equation 14.

The net effect of our base distribution is to bias the model towards simple rules with a small number of internal nodes. The geometric increase in cost associated with the stopping decisions discourages the model from using larger rules; for these rules to be included they must occur very frequently in the corpus. Similarly, rules which use high-probability (frequent) CFG productions are favoured. It is unclear if these biases are ideal: we anticipate that other, more sophisticated distributions would improve the model's performance.

In the unsupervised setting we no longer have a training set of annotated trees and therefore do not have a PCFG readily available to use as the base distribution in Equation 7. For this reason we extend the previous model to a two level hierarchy of PYPs. As before, the topmost level is defined over the elementary tree fragments (G_c) with the base distribution (P_{E}) assigning probability to the infinite space of possible fragments. The model differs from the supervised one by defining P_C in (7) using a PYP prior over CFG rules. Accordingly the model can now infer a two level hierarchy consisting of a PCFG embedded within a TSG, compared to the supervised parsing model which only learnt the TSG level with a fixed PCFG. Formally, each CFG production is drawn from[7]

$$\begin{aligned} H_c|\, a'_c, b'_c &\sim \mathrm{PYP}(a'_c, b'_c, \mathrm{Uniform}(\cdot|c)) \\ \alpha|\, c, H_c &\sim H_c \,, \end{aligned} \tag{8}$$

7. As we are using a finite base distribution over CFG productions, we could use a Dirichlet instead of the PYP presented in (8). However we elect to use a PYP because it is more general, having additional expressive power from its discounting behaviour.

where a'_c and b'_c are the PYP hyper-parameters and Uniform$(\cdot|c)$ is a uniform distribution over the space of rewrites for non-terminal c.[8] As before, we integrate out the model parameters, H_c. Consequently draws from P_C are no longer *iid* but instead are tied in the prior, and the probability of the sequence of component CFG productions $\{c' \to \alpha \in e\}$ now follows a Pitman-Yor Chinese Restaurant Process.

The CFG level and TSG level PYCRPs are connected as follows: every time an elementary tree is assigned to a new table in the TSG level, each of its component CFG rules are drawn from the CFG level prior. Note that, just as elementary trees are divided into separate restaurants at the TSG level based on their root categories, CFG rules are divided into separate restaurants at the CFG level based on their left-hand sides. Formally, the probability of r_j, the j^{th} CFG rule in the sequence, is given by

$$P_C(r_j = r | c_j = c, \mathbf{z}'_{-j}, \ell'_{-j}) = \frac{n'^{-}_r - K'^{-}_r a'_c + (K'^{-}_c a'_c + b'_c)\frac{1}{|R_c|}}{K'^{-}_c + b'_c}, \tag{9}$$

where c_j is the left-hand side of r_j; $\mathbf{z}'_{-j}$ and ℓ'_{-j} are the table assignments and table labels in the CFG-level restaurants (we use prime symbols to indicate variables pertaining to the CFG level); n'^{-}_r is the number of times rule r is used in any table label in a TSG restaurant (equivalently, the number of customers at tables labelled r in the CFG restaurants); K'^{-}_r and $K'^{-}_c = \sum_{r:\text{root}(r)=c} K'^{-}_r$ are the CFG-level table counts for r and all rules rewriting c, respectively; and R_c is the set of CFG productions which can rewrite c. This formulation reflects that we now have multiple tied restaurants, and each time an elementary tree opens a new table in a top-level restaurant all its rules are considered to have entered their own respective P_C restaurants (according to their root c). Accordingly the CFG-level customer count, n'^{-}_r, is the number of occurrences of r in the elementary trees that label the tables in the TSG restaurants (excluding r_j). Thus, in the unsupervised case, the product of rule probabilities (the final factor) in Equation (7) is computed by multiplying together the conditional probability of each rule (9) given the previous ones.

4. Training

We present two alternative algorithms for training our model, both based on Markov chain Monte Carlo techniques, which produce samples from the posterior distribution of the model by iteratively resampling the values of the hidden variables (tree nodes). The first algorithm is a *local* sampler, which operates by making a local update to a single tree node in each sampling step. The second algorithm is a *blocked* sampler, which makes much larger moves by sampling analyses for full sentences, which should improve the mixing over the local sampler. Importantly the blocked sampler is more general, being directly applicable to both supervised and unsupervised settings (and for parsing test sentences, which is equivalent to an unsupervised setting) while the local sampler is only applicable for supervised learning, where the trees are observed. We now present the two sampling methods in further detail.

8. In our experiments on unsupervised dependency parsing the space of rewrites varied depending on c, and can be as large as the set of part-of-speech tags. See Section 7 for details.

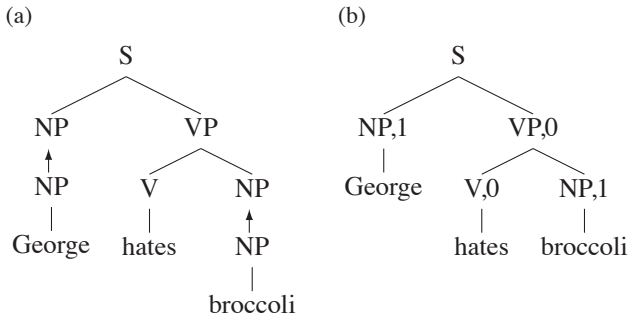


Figure 5: Gibbs sampler state (b) corresponding to the example derivation (a) (reproduced from Figure 1a). Each node is labelled with its substitution variable.

4.1 Local Sampler

The *local* sampler is designed specifically for the supervised scenario, and samples a TSG derivation for each tree by sampling local updates at each tree node. It uses Gibbs sampling (Geman and Geman, 1984), where random variables are repeatedly sampled conditioned on the current values of all other random variables in the model. The actual algorithm is analogous to the Gibbs sampler used for inference in the Bayesian model of word segmentation presented by Goldwater et al. (2006); indeed, the problem of inferring the derivations $\mathbf{e}$ from $\mathbf{t}$ can be viewed as a segmentation problem, where each full tree must be segmented into one or more elementary trees. To formulate the local sampler, we associate a binary variable $x_d \in \{0,1\}$ with each non-root internal node, d, of each tree in the training set, indicating whether that node is a substitution point ($x_d = 1$) or not ($x_d = 0$). Each substitution point forms the root of some elementary tree, as well as a frontier nonterminal of an ancestor node's elementary tree. Conversely, each non-substitution point forms an internal node inside an elementary tree. Collectively the training trees and substitution variables specify the sequence of elementary trees $\mathbf{e}$ that is the current state of the sampler. Figure 5 shows an example tree with its substitution variables and its corresponding TSG derivation.

Our Gibbs sampler works by sampling the value of the x_d variables, one at a time, in random order. If d is the node associated with x_d, the substitution variable under consideration, then the two possible values of x_d define two options for $\mathbf{e}$: one in which d is internal to some elementary tree e_M, and one in which d is the substitution site connecting two smaller trees, e_A and e_B. In the example in Figure 5, when sampling the VP node, $e_M =$ (S NP (VP (V hates) NP)), $e_A =$ (S NP VP), and $e_B =$ (VP (V hates) NP). To sample a value for x_d, we compute the probabilities of e_M and (e_A, e_B), conditioned on $\mathbf{e}^-$: all other elementary trees in the training set that share at most a root or frontier nonterminal with e_M, e_A, or e_B. These probabilities are easy to compute because the PYP is *exchangeable*, meaning that the probability of a set of outcomes does not depend on their ordering. Therefore we can treat the elementary trees under consideration as the last ones to be sampled, and apply Equation (6). We then sample one of the two outcomes (merging or splitting) according to the relative probabilities of these two events. More specifically, the probabilities of the two outcomes,

conditioned on the current analyses of the remainder of the corpus, are

$$P(e_M|c_M) = \frac{n^-_{e_M} - K^-_{e_M} a_{c_M} + (K^-_{c_M} a_{c_M} + b_{c_M}) P_{\mathrm{E}}(e_M|c_M)}{n^-_{c_M} + b_{c_M}} \quad \text{and}$$

$$\begin{aligned} P(e_A, e_B|c_A, c_B) &= \sum_{z_{e_A}} P(e_A, z_{e_A}|c_A) P(e_B|e_A, z_{e_A}, c_A, c_B) \\ &= \frac{n^-_{e_A} - K^-_{e_A} a_{c_A}}{n^-_{c_A} + b_{c_A}} \times \frac{n^-_{e_B} + \delta_e - K^-_{e_B} a_{c_B} + (K^-_{c_B} a_{c_B} + b_{c_B}) P_{\mathrm{E}}(e_B|c_B)}{n^-_{c_B} + \delta_c + b_{c_B}} \\ &\quad + \frac{(K^-_{c_A} a_{c_A} + b_{c_A}) P_{\mathrm{E}}(e_A|c_A)}{n^-_{c_A} + b_{c_A}} \\ &\quad \times \frac{n^-_{e_B} + \delta_e - (K^-_{e_B} + \delta_e) a_{c_B} + \left((K^-_{c_B} + \delta_c) a_{c_B} + b_{c_B}\right) P_{\mathrm{E}}(e_B|c_B)}{n^-_{c_B} + \delta_c + b_{c_B}}, \end{aligned} \tag{10}$$

where c_y is the root label of e_y, the counts n^- and K^- are derived from $\mathbf{z}_{-M}$ and $\ell(\mathbf{z}_{-M})$ (this dependency is omitted from the conditioning context for brevity), $\delta_e = \delta(e_A, e_B)$ is the Kronecker delta function which has value one when e_A and e_B are identical and zero otherwise, and similarly for $\delta_c = \delta(c_A, c_B)$ which compares their root nonterminals c_A and c_B. The δ terms reflect the changes to n^- that would occur after observing e_A, which forms part of the conditioning context for e_B. The two additive terms in (10) correspond to different values of z_{e_A}, the seating assignment for e_A. Specifically, the first term accounts for the case where e_A is assigned to an existing table, $z_{e_A} < K^-_{e_A}$, and the second term accounts for the case where e_A is seated at a new table, $z_{e_A} = K^-_{e_A}$. The seating affects the conditional probability of e_B by potentially increasing the number of tables $K^-_{e_A}$ or $K^-_{c_A}$ (relevant only when $e_A = e_B$ or $c_A = c_B$).

4.2 Blocked Sampler

The local sampler has the benefit of being extremely simple, however it may suffer from slow convergence (poor mixing) due to its very local updates. That is, it can get stuck because many locally improbable decisions are required to escape from a locally optimal solution. Moreover it is only applicable to the supervised setting: it cannot be used for unsupervised grammar induction or for parsing test strings. For these reasons we develop the *blocked* sampler, which updates blocks of variables at once, where each block consists of all the the nodes associated with a single sentence. This sampler can make larger moves than the local sampler and is more flexible, in that it can perform inference with both string (unsupervised) or tree (supervised) input.[9]

The blocked sampler updates the analysis for each sentence given the analyses for all other sentences in the training set. We base our approach on the algorithm developed by Johnson et al. (2007a) for sampling parse trees using a finite Bayesian PCFG model with Dirichlet priors over the multinomial rule probabilities. As in our model, they integrate out the parameters (in their case, the PCFG rule probabilities), leading to a similar caching effect due to interdependences between the latent variables (PCFG rules in the parse). Thus, standard dynamic programming algorithms cannot be used to sample directly from the desired posterior, $p(t|\mathbf{w}, \mathbf{t}^-)$, that is, the distribution of parse trees for the current sentence given the words in the corpus and the trees for all other sentences. To solve this problem, Johnson et al. (2007a) developed a Metropolis-Hastings (MH)

9. A recently-proposed alternative approach is to perform *type-level* updates, which samples updates to many similar tree fragments at once (Liang et al., 2010). This was shown to converge faster than the local Gibbs sampler.

sampler. The MH algorithm is an MCMC technique which allows for samples to be drawn from a probability distribution, $\pi(s)$, by first drawing samples from a *proposal distribution*, $Q(s'|s)$, and then correcting these to the true distribution using an acceptance/rejection step. Given a state s, we sample a next state $s' \sim Q(\cdot|s)$ from the proposal distribution; this new state is accepted with probability

$$A(s,s') = \min\left\{\frac{\pi(s')Q(s|s')}{\pi(s)Q(s'|s)}, 1\right\}$$

and is rejected otherwise, in which case s is retained as the current state. The Markov chain defined by this process is guaranteed to converge on the desired distribution, $\pi(s)$. Critically, the MH algorithm enables sampling from distributions from which we cannot sample directly, and moreover, we need not know the normalisation constant for $\pi(\cdot)$, since it cancels in $A(s,s')$.

In Johnson et al.'s (2007a) algorithm for sampling from a Bayesian PCFG, the proposal distribution is simply $Q(t'|t) = P(t'|\theta^{MAP})$, the posterior distribution over trees given fixed parameters θ^{MAP}, where θ^{MAP} is the MAP estimate based on the conditioning data, $\mathbf{t}^{-}$. Note that the proposal distribution is a close fit to the true posterior, differing only in that under the MAP the production probabilities in a derivation are *iid*, while for the true model the probabilities are tied by the prior (giving rise to the caching effect). The benefit of using the MAP is that its independences mean that inference can be solved using dynamic programming, namely the inside algorithm (Lari and Young, 1990). Given the inside chart, which stores the aggregate probability of all subtrees for each word span and rooted with a given nonterminal label, samples can be drawn using a simple top-down recursion (Johnson et al., 2007a).

Our model is similar to Johnson et al.'s, as we also use a Bayesian prior in a model of grammar induction and consequently face similar problems with directly sampling due to the caching effects of the prior. For this reason, we use the MH algorithm in a similar manner, except in our case we draw samples of derivations of elementary trees and their seating assignments, $p(\mathbf{z}_i, \ell_i|\mathbf{w}, \mathbf{z}_{-i}, \ell_{-i})$, and use a MAP estimate over $(\mathbf{z}_{-i}, \ell_{i-})$ as our proposal distribution.[10] However, we have an added complication: the MAP cannot be estimated directly. This is a consequence of the base distribution having infinite support, which means the MAP has an infinite rule set. For finite TSG models, such as those used in DOP, constructing a CFG equivalent grammar is straightforward (if unwieldy). This can be done by creating a rule for each elementary tree which rewrites its root nonterminal as its frontier. For example under this technique S $\rightarrow$ NP (VP (V hates) NP would be mapped to S $\rightarrow$ NP hates NP.[11] However, since our model has infinite support over productions, it cannot be mapped in the same way. For example, if our base distribution licences the CFG production NP $\rightarrow$ NP PP then our TSG grammar will contain the infinite set of elementary trees NP $\rightarrow$ NP PP, NP $\rightarrow$ (NP NP PP) PP, NP $\rightarrow$ (NP (NP NP PP) PP) PP, ..., each with decreasing but non-zero probability. These would all need to be mapped to CFG rules in order to perform inference under the grammar, which is clearly impossible.

Thankfully it is possible to transform the infinite MAP-TSG into a finite CFG, using a method inspired by Goodman (2003), who developed a grammar transform for efficient parsing with an

10. Calculating the proposal and acceptance probabilities requires sampling not just the elementary trees, but also their table assignments (for both levels in the hierarchical model). We elected to simplify the implementation by separately sampling the elementary trees and their table assignments.

11. Alternatively, interspersing a special nonterminal, for example, S $\rightarrow$ {S-NP-{VP-{V-hates}-NP} $\rightarrow$ NP hates NP, encodes the full structure of the elementary tree, thereby allowing the mapping to be reversed. We use a similar technique to encode non-zero count rules in our grammar transformation, described below.

all-subtrees DOP grammar. In the transformed grammar inside inference is tractable, allowing us to draw proposal samples efficiently and thus construct a Metropolis-Hastings sampler. The resultant grammar allows for efficient inference, both in unsupervised and supervised training and in parsing (see Section 5).

We represent the MAP using the grammar transformation in Table 1, which separates the count and base distribution terms in Equation 6 into two separate CFGs, denoted A and B. We reproduce Equation 6 below along with its decomposition:

$$\begin{aligned} P(e_i = e \,|\, c, \mathbf{z}_{-i}, \ell(\mathbf{z}_{-i})) &= \frac{n_e^- - K_e^- a_c + (K_c^- a_c + b_c) P_{\text{E}}(e|c)}{n_c^- + b_c} \\ &= \underbrace{\frac{n_e^- - K_e^- a_c}{n_c^- + b_c}}_{\text{count}} + \underbrace{\frac{K_c^- a_c + b_c}{n_c^- + b_c} P_{\text{E}}(e|c)}_{\text{base}} . \end{aligned} \tag{11}$$

Grammar A has productions for every elementary tree e with $n_e^- \geq 1$, which are assigned as their probability the count term in Equation 11.[12] The function $\text{sig}(e)$ returns a string signature for elementary trees, for which we use a form of bracketed notation. To signify the difference between these nonterminal symbols and trees, we use curly braces and hyphens in place of round parentheses and spaces, respectively, for example, the elementary tree (S NP (VP (V hates) NP)) is denoted by the nonterminal symbol {S-NP-{VP-{V-hates}-NP}}. Grammar B has productions for every CFG production licensed under P_{E}; its productions are denoted using primed (') nonterminals. The rule $c \to c'$ bridges from A to B, weighted by the base term in Equation 11 excluding the $P_{\text{E}}(e|c)$ factor. The contribution of the base distribution is computed recursively via child productions. The remaining rules in grammar B correspond to every CFG production in the underlying PCFG base distribution, coupled with the binary decision of whether or not nonterminal children should be substitution sites (frontier nonterminals). This choice affects the rule probability by including an s or $1-s$ factor, and child substitution sites also function as a bridge back from grammar B to A. There are often two equivalent paths to reach the same chart cell using the same elementary tree—via grammar A using observed TSG productions and via grammar B using P_{E} backoff—which are summed to yield the desired net probability. The transform is illustrated in the example in Figures 6 and 7.

Using the transformed grammar we can represent the MAP grammar efficiently and draw samples of TSG derivations using the inside algorithm. In an unsupervised setting, that is, given a yield string as input, the grammar transform above can be used directly with the inside algorithm for PCFGs (followed by the reverse transform to map the sampled derivation into TSG elementary trees). This has an asymptotic time complexity cubic in the length of the input.

For supervised training the trees are observed and thus we must ensure that the TSG analysis matches the given tree structure. This necessitates constraining the inside algorithm to only consider spans that are present in the given tree and with the given nonterminal. Nonterminals are said to match their primed and signed counterparts, for example, VP′ and {VP-{V-hates}-NP} both match VP. A sample from the constrained inside chart will specify the substitution variables for each node in the tree: For each node d if it has a non-primed category in the sample then it is a substitution

12. The transform assumes inside inference, where alternate analyses for the same span of words with the same nonterminal are summed together. In Viterbi inference the summation is replaced by maximisation, and therefore we need different expansion probabilities. This requires changing the weight for $c \to \text{sig}(e)$ to $P(e_i = e \,|\, c, \mathbf{z}_{-i}, \ell(\mathbf{z}_{-i}))$ in Table 1.

Grammar A	For every ET, e, rewriting c with non-zero count:	
	$c \to \text{sig}(e)$	$\frac{n_e^- - K_e^- a_c}{n_c^- + b_c}$
	For every internal node e_i in e with children $e_{i,1}, \ldots, e_{i,n}$	
	$\text{sig}(e_i) \to \text{sig}(e_{i,1}) \ldots \text{sig}(e_{i,n})$	1
A $\to$ B	For every nonterminal, c:	
	$c \to c'$	$\frac{K_c^- a_c + b_c}{n_c^- + b_c}$
Grammar B	For every pre-terminal CFG production, $c \to t$:	
	$c' \to t$	$P_{\text{C}}(c \to t)$
	For every unary CFG production, $c \to a$:	
	$c' \to a$	$P_{\text{C}}(c \to a) s_a$
	$c' \to a'$	$P_{\text{C}}(c \to a)(1 - s_a)$
	For every binary CFG production, $c \to ab$:	
	$c' \to ab$	$P_{\text{C}}(c \to ab) s_a s_b$
	$c' \to ab'$	$P_{\text{C}}(c \to ab) s_a (1 - s_b)$
	$c' \to a'b$	$P_{\text{C}}(c \to ab)(1 - s_a) s_b$
	$c' \to a'b'$	$P_{\text{C}}(c \to ab)(1 - s_a)(1 - s_b)$

Table 1: Grammar transformation rules to map an infinite MAP TSG into an equivalent CFG, separated into three groups for grammar A (top), the bridge between A $\to$ B (middle) and grammar B (bottom). Production probabilities are shown to the right of each rule. The $\text{sig}(e)$ function creates a unique string signature for an ET e (where the signature of a frontier node is itself) and s_c is the probability of c being a substitution variable, thus stopping the P_{E} recursion.

S $\to$ {NP-{VP-{V-hates}-NP}}	$\frac{n_e^- - K_e^- a_S}{n_S^- + b_S}$
{NP-{VP-{V-hates}-NP}} $\to$ NP {VP-{V-hates}-NP}	1
{VP-{V-hates}-NP} $\to$ {V-hates} NP	1
{V-hates} $\to$ hates	1
S $\to$ S'	$\frac{K_S^- a_S + b_S}{n_S^- + b_S}$
S' $\to$ NP VP'	$P_{\text{C}}(\text{S} \to \text{NP VP}) s_{NP} (1 - s_{VP})$
VP' $\to$ V' NP	$P_{\text{C}}(\text{VP} \to \text{V NP})(1 - s_V) s_{NP}$
V' $\to$ hates	$P_{\text{C}}(\text{V} \to \text{hates})$

Figure 6: Example showing the transformed grammar rules for the single elementary tree $e =$ (S NP (VP (V hates) NP)) and the scores for each rule. Only the rules which correspond to e and its substitution sites are displayed. Taking the product of the rule scores above the dashed line yields the *count* term in (11), and the product of the scores below the line yields the *base* term. When the two analyses are combined and their probabilities summed together, we get $P(e_i = e | c, \mathbf{z}_{-i}, \ell(\mathbf{z}_{-i}))$.

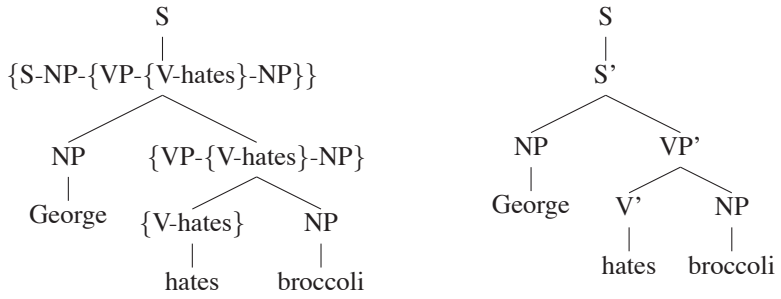


Figure 7: Example trees under the grammar transform, which both encode the same TSG derivation from Figure 1a. The left tree encodes that the S $\rightarrow$ NP (VP (V hates) NP elementary tree was drawn from the cache, while for the right tree this same elementary tree was drawn from the base distribution (the count and base terms in (11), respectively).

site, $x_d = 1$, otherwise it is an internal node, $x_d = 0$. For example, both trees in Figure 7 encode that both NP nodes are substitution sites and that the VP and V nodes are not substitution sites (the same configuration as Figure 5).

The time complexity of the constrained inside algorithm is linear in the size of the tree and the length of the sentence. The local sampler also has the same time complexity, however it is not immediately clear which technique will be faster in practise. It is likely that the blocked sampler will have a slower runtime due to its more complicated implementation, particularly in transforming the grammar and inside inference. Although the two samplers have equivalent asymptotic complexity, the constant factors may differ greatly. In Section 6 we compare the two training methods empirically to determine which converges more quickly.

4.3 Sampling Hyperparameters

In the previous discussion we assumed that we are given the model hyperparameters, $(\mathbf{a}, \mathbf{b}, \mathbf{s})$. While it might be possible to specify their values manually or fit them using a development set, both approaches are made difficult by the high dimensional parameter space. Instead we treat the hyperparameters as random variables in our model, by placing vague priors over them and infer their values during training. This is an elegant way to specify their values, although it does limit our ability to tune the model to optimise a performance metric on held-out data.

For the PYP discount parameters $\mathbf{a}$, we employ independent Beta priors, $a_c \sim \text{Beta}(1,1)$. The prior is uniform, encoding that we have no strong prior knowledge of what the value of each a_c should be. The conditional probability of a_c given the current derivations $\mathbf{z}, \ell$ is

$$P(a_c|\mathbf{z},\ell) \propto P(\mathbf{z},\ell|a_c) \times \text{Beta}(a_c|1,1)\,.$$

We cannot calculate the normaliser for this probability, however $P(\mathbf{z},\ell|a_c)$ can be calculated using Equation 3 and thus $P(a_c|\mathbf{z},\ell)$ can be calculated up to a constant. We use the range doubling slice sampling technique of Neal (2003) to draw a new sample of a_c' from its conditional distribution.[13]

We treat the concentration parameters, $\mathbf{b}$, as being generated by independent gamma priors, $b_c \sim \text{Gamma}(1,1)$. We use the same slice-sampling method for a_c to sample from the conditional

13. We used the slice sampler included in Mark Johnson's Adaptor Grammar implementation, available at `http://web.science.mq.edu.au/~mjohnson/Software.htm`.

over b_c,

$$P(b_c|\mathbf{z},\ell) \propto P(\mathbf{z},\ell|b_c) \times \text{Gamma}(b_c|1,1)\,.$$

This prior is not vague, that is, the probability density function decays exponentially for higher values of b_c, which serves to limits the influence of the P_{C} prior. In our experimentation we found that this bias had little effect on the generalisation accuracy of the supervised and unsupervised models, compared to a much vaguer Gamma prior with the same mean.

We use a vague Beta prior for the stopping probabilities in P_{E}, $s_c \sim \text{Beta}(1,1)$. The Beta distribution is conjugate to the binomial, and therefore the posterior is also a Beta distribution from which we can sample directly,

$$s_c \sim \text{Beta}\left(1+\sum_e K_e(\mathbf{z}) \sum_{n\in F(e)} \delta(n,c),\, 1+\sum_e K_e(\mathbf{z}) \sum_{n\in I(e)} \delta(n,c)\right),$$

where e ranges over all non-zero count elementary trees, $F(e)$ are the nonterminal frontier nodes in e, $I(e)$ are the non-root internal nodes and the δ terms count the number of nodes in e with nonterminal c. In other words, the first Beta argument is the number of tables in which a node with nonterminal c is a stopping node in the P_{E} expansion and the second argument is the number of tables in which c has been expanded (a non-stopping node).

All the hyper-parameters are resampled after every full sampling iteration over the training trees, except in the experiments in Section 7 where they are sampled every 10th iteration.

5. Parsing

We now turn to the problem of using the model to parse novel sentences. This requires finding the maximiser of

$$p(t|\omega,\mathbf{w}) = \int\int\int\int p(t|\omega,\mathbf{z},\ell,\mathbf{a},\mathbf{b},\mathbf{s})\, p(\mathbf{z},\ell,\mathbf{a},\mathbf{b},\mathbf{s}|\mathbf{w})\, d\mathbf{z}\, d\ell\, d\mathbf{a}\, d\mathbf{b}\, d\mathbf{s}\,, \tag{12}$$

where ω is the sequence of words being parsed, t is the resulting tree, $\mathbf{w}$ are the training sentences, $\mathbf{z}$ and ℓ represent their parses, elementary tree representation and table assignments and $(\mathbf{a},\mathbf{b},\mathbf{s})$ are the model's hyper-parameters. For the supervised case we use the training trees, $\mathbf{t}$, in place of $\mathbf{w}$ in Equation 12.

Unfortunately solving for the maximising parse tree in Equation 12 is intractable. However, it can be approximated using Monte Carlo techniques. Given a sample of $(\mathbf{z},\ell,\mathbf{a},\mathbf{b},\mathbf{s})$ we can reason over the space of possible TSG derivations, e, for sentence w using the same Metropolis-Hastings sampler presented in Section 4.2 for blocked inference in the unsupervised setting.[14] This gives us a set of samples from the posterior $p(\mathbf{e}|w,\mathbf{z},\ell,\mathbf{a},\mathbf{b},\mathbf{s})$. We then use a Monte Carlo integral to obtain a marginal distribution over trees (Bod, 2003),

$$\hat{p}^{MPT}(t) = \sum_{m=1}^{M} \delta(t, \text{tree}(\mathbf{e}_m))\,, \tag{13}$$

14. Using many samples in a Monte Carlo integral is a straight-forward extension to our parsing algorithm. We did not observe a significant improvement in parsing accuracy when using a multiple samples compared to a single sample, and therefore just present results for a single sample. However, using multiple models has been shown to improve the performance of other parsing models (Petrov, 2010).

where $\{\mathbf{e}_m\}_{m=1}^M$ are our sample of derivations for w. It is then straightforward to find the best parse, $t^* = \arg\max \hat{p}(t)$, which is simply the most frequent tree in the sample.

In addition to solving from the maximum probability tree (MPT) using Equation 13, we also present results for a number of alternative objectives. To test whether the derivational ambiguity is important, we also compute the maximum probability derivation (MPD),

$$\hat{p}^{MPD}(\mathbf{e}) = \sum_{m=1}^{M} \delta(\mathbf{e}, \mathbf{e}_m),$$

using a Monte-Carlo integral, from which we recover the tree, $t^* = \text{tree}(\arg\max_{\mathbf{e}} \hat{p}^{MPD}(\mathbf{e}))$. We also compare using the Viterbi algorithm directly with the MAP grammar, $t^* = \text{tree}(\arg\max_{\mathbf{e}} P_{MAP}(\mathbf{e}|w))$, which constitutes an approximation to the true model in which we can search exactly. This contrasts with the MPD which performs approximate search under the true model. We compare the different methods empirically in Section 6.

The MPD and MPT parsing algorithms require the computation of Monte-Carlo integrals over the large space of possible derivations or trees. Consequently, unless the distribution is extremely peaked the chance of sampling many identical structures is small, vanishingly so for long sentences (the space of trees grows exponentially with the sentence length). In other words, the sampling variance can be high which could negatively affect parsing performance. For this reason we present an alternative parsing method which compiles more local statistics for which we can obtain reliable estimates. The technique is based on Goodman's (2003) algorithm for maximising labelled recall in DOP parsing and subsequent work on parsing in state-splitting CFGs (Petrov and Klein, 2007). The first step is to acquire marginal distributions over the CFG productions within each sampled tree. Specifically, we collect counts for events of the form $(c \to \alpha, i, j, k)$, where $c \to \alpha$ is a CFG production spanning words $[i, j)$ and k is the split point between child constituents for binary productions, $i < k < j$ ($k = 0$ for unary productions). These counts are then marginalised by the number of trees sampled. Finally the Viterbi algorithm is used to find the tree with the maximum cumulative probability under these marginals, which we call the maximum expected rule (MER) parse. Note that this is a type of minimum Bayes risk decoding and was first presented in Petrov and Klein (2007) as the MAX-RULE-SUM method (using exact marginals, not Monte-Carlo estimates as is done here).

6. Supervised Parsing Experiments

In this section we present an empirical evaluation of the model on the task of supervised parsing. In this setting the model learns a segmentation of a training treebank, which defines a TSG. We present parsing results using the learned grammar, comparing the effects of the sampling strategy, initialisation conditions, parsing algorithm and the size of the training set. The unsupervised model is described in the following section, Section 7.

We trained the model on the WSJ part of Penn. treebank (Marcus et al., 1993) using the standard data splits, as shown in Table 2. As our model is parameter free (the $\mathbf{a}, \mathbf{b}, \mathbf{s}$ parameters are learnt in training), we do not use the development set for parameter tuning. We expect that fitting the hyperparameters to maximise performance on the development set would lead to a small increase in generalisation performance, but at a significant cost in runtime. We adopt Petrov et al.'s (2006) method for data preprocessing: right-binarizing the trees to limit the branching factor and replacing tokens with count ≤ 1 in the training sample with one of roughly 50 generic unknown word markers which convey the token's lexical features and position. The predicted trees are evaluated using

Partition	sections	sentences	tokens	types	types (unk)
training	2–21	33180	790237	40174	21387
development	22	1700	40117	6840	5473
testing	23	2416	56684	8421	6659
small training	2	1989	48134	8503	3728

Table 2: Corpus statistics for supervised parsing experiments using the Penn treebank, reporting for each partition its WSJ section/s, the number of sentences, word tokens and unique word types. The final column shows the number of word types after unknown word processing using the full training set, which replaces rare words with placeholder tokens. The number of types after preprocessing in the development and testing sets is roughly halved when using the the small training set.

EVALB[15] and we report the F1 score over labelled constituents and exact match accuracy over all sentences in the testing sets.

In our experiments, we initialised the sampler by setting all substitution variables to 0, thus treating every full tree in the training set as an elementary tree. Unless otherwise specified, the blocked sampler was used for training. We later evaluated the effect of different starting conditions on the quality of the configurations found by the sampler and on parsing accuracy. The sampler was trained for 5000 iterations and we use the final sample of $\mathbf{z}, \ell, \mathbf{a}, \mathbf{b}, \mathbf{s}$ for parsing. We ran all four different parsing algorithms and compare their results on the testing sets. For the parsing methods that require a Monte Carlo integral (MPD, MPT and MER), we sampled 1000 derivations from the MAP approximation grammar which were then input to the Metropolis-Hastings acceptance step before compiling the relevant statistics. The Metropolis-Hastings acceptance rate was around 99% for both training and parsing. Each experiment was replicated five times and the results averaged.

6.1 Small Data Sample

For our first treebank experiments we train on a small data sample by using only section 2 of the treebank (see Table 2 for corpus statistics.) Bayesian methods tend to do well with small data samples, while for larger samples the benefits diminish relative to point estimates. For this reason we present a series of exploratory experiments on the small data set before moving to the full treebank.

In our experiments we aim to answer the following questions: Firstly, in terms of parsing accuracy, does the Bayesian TSG model outperform a PCFG baseline, and how does it compare to existing high-quality parsers? We will also measure the effect of the parsing algorithm: Viterbi, MPD, MPT and MER. Secondly, which of the local and blocked sampling techniques is more efficient at mixing, and which is faster per iteration? Finally, what kind of structures does the model learn and do they match our expectations? The hyper-parameter values are also of interest, particularly to evaluate whether the increased generality of the PYP is justified over the DP. Our initial experiments aim to answer these questions on the small data set, after which we take the best model and apply it to the full set.

Table 3 presents the prediction results on the development set. The baseline is a maximum likelihood PCFG. The TSG models significantly outperform the baseline. This confirms our hypothesis that CFGs are not sufficiently powerful to model syntax, and that the increased context afforded to

15. See `http://nlp.cs.nyu.edu/evalb/`.

Model	Viterbi	MPD	MPP	MER	# rules
PCFG	60.20	60.20	60.20	-	3500
TSG PYP	74.90	76.68	77.17	78.59	25746
TSG DP	74.70	75.86	76.24	77.91	25339
Berkeley parser ($\tau = 2$)	71.93	71.93	-	74.32	16168
Berkeley parser ($\tau = 5$)	75.33	75.33	-	77.93	39758

Table 3: Development results for models trained on section 2 of the Penn treebank, showing labelled constituent F1 and the grammar size. For the TSG models the grammar size reported is the number of CFG productions in the transformed MAP PCFG approximation. Unknown word models are applied to words occurring less than two times (TSG models and Berkeley $\tau = 2$) or less than five times (Berkeley $\tau = 5$).

the TSG can make a large difference. Surprisingly, the MPP technique is only slightly better than the MPD approach, suggesting that derivational ambiguity is not as much of a problem as previously thought (Bod, 2003). Also surprising is the fact that exact Viterbi parsing under the MAP approximation is much worse than the MPD method which uses an approximate search technique under the true model. The MER technique is a clear winner, however, with considerably better F1 scores than either MPD or MPP, with a margin of 1–2 points. This method is less affected by sampling variance than the other MC algorithms due to its use of smaller tree fragments (CFG productions at each span).

We also consider the difference between using a Dirichlet process prior (DP) and a Pitman-Yor process prior (PYP). This amounts to whether the $\mathbf{a}$ hyper-parameters are set to 0 (DP) or are allowed to take on non-zero values (PYP), in which case we sample their values as described in Section 4.3. There is a small but consistent gain of around 0.5 F1 points across the different parsing methods from using the PYP, confirming our expectation that the increased flexibility of the PYP is useful for modelling linguistic data. Figure 8a shows the learned values of the PYP hyperparameters after training for each nonterminal category. It is interesting to see that the hyper-parameter values mostly separate the open-class categories, which denote constituents carrying semantic content, from the closed-class categories, which are largely structural. The open classes (noun-, verb-, adjective- and adverb-phrases: NP, VP, ADJP and ADVP, respectively) tend to have higher a and b values (towards the top right corner of the graph) and therefore can describe highly diverse sets of productions. In contrast, most of the closed classes (the root category, quantity phrases, wh-question noun phrases and sentential phrases: TOP, QP, WHNP and S, respectively) have low a and b (towards the bottom left corner of the graph), reflecting that encoding their largely formulaic rewrites does not necessitate diverse distributions.

The $\mathbf{s}$ hyper-parameter values are shown in Figure 8b, and are mostly in the mid-range (0.3–0.7). Prepositions (IN), adverbs (RB), determiners (DT) and some tenses of verbs (VBD and VBP) have very low s values, and therefore tend to be lexicalized into elementary trees. This is expected behaviour, as these categories select strongly for the words they modify and some (DT, verbs) must agree with their arguments in number and tense. Conversely particles (RP), modal verbs (MD) and possessive particles (PRP$) have high s values, and are therefore rarely lexicalized. This is reasonable for MD and PRP$, which tend to be exchangeable for one another without rendering the sentence ungrammatical (e.g., 'his' can be substituted for 'their' and 'should' for 'can'). However,

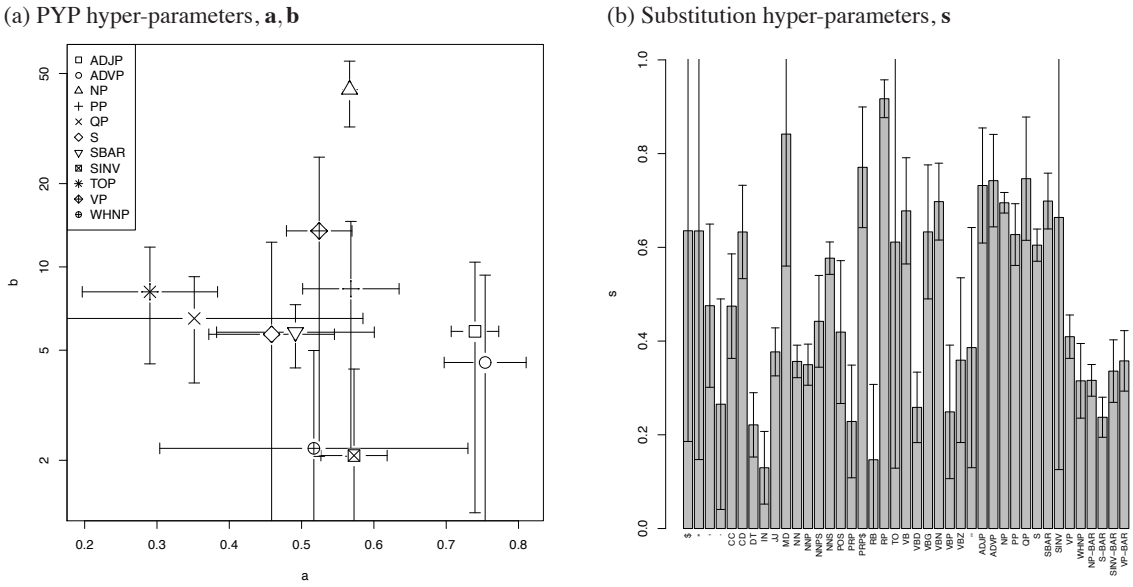


Figure 8: Inferred hyper-parameter values. Points (left) or bars (right) show the mean value with error bars indicating one standard deviation, computed over the final samples of five sampling runs. For legibility, a) has been limited to common phrasal nonterminals, while b) also shows preterminals and binarized nonterminals (those with the -BAR suffix). Note that in a) b is plotted on a log-scale.

particles are highly selective for the verbs they modify, and therefore should be lexicalized by the model (e.g., for 'tied in', we cannot substitute 'out' for 'in'). We postulate that the model does not learn to lexicalise verb-particle constructions because they are relatively uncommon, they often occur with different tenses of verb and often the particle is not adjacent to the verb, therefore requiring large elementary trees to cover both words. The phrasal types all have similar s values except for VP, which is much more likely to be lexicalized. This allows for elementary trees combining a verb with its subject noun-phrase, which is typically not part of the VP, but instead a descendent of its parent S node. Finally, the s values for the binarized nodes (denoted with the -BAR suffix) on the far right of Figure 8b are all quite low, encoding a preference for the model to reconstitute binarized productions into their original form. Some of the values have very high variance, for example, \$, which is due to their rewriting as a single string with probability 1 under P_C (or a small set of strings and a low entropy distribution), thus making the hyper-parameter value immaterial.

For comparison, we trained the Berkeley split-merge parser (Petrov et al., 2006) on the same data and decoded using the Viterbi algorithm (MPD) and expected rule count (MER, also known as MAX-RULE-SUM). We ran two iterations of split-merge training, after which the development F1 dropped substantially (in contrast, our model is not fit to the development data). The result (denoted $\tau = 2$) is an accuracy slightly below that of our model. To be fairer to their model, we adjusted the unknown word threshold to their default setting, that is, to apply to word types occurring fewer than five times (denoted $\tau = 5$).[16] Note that our model's performance degraded using the higher

16. The Berkeley parser has a number of further enhancements that we elected not to use, most notably, a more sophisticated means of handling unknown words. These enhancements produce further improvements in parse accuracy, but could also be implemented in our model to similar effect.

	≤ 40		all	
Parser	**F1**	**EX**	**F1**	**EX**
MLE PCFG	64.2	7.2	63.1	6.7
TSG PYP Viterbi	83.6	24.6	82.7	22.9
TSG PYP MPD	84.2	27.2	83.3	25.4
TSG PYP MPT	84.7	28.0	83.8	26.2
TSG PYP MER	85.4	27.2	84.7	25.8
DOP (Zuidema, 2007)			83.8	26.9
Berkeley parser (Petrov and Klein, 2007)	90.6		90.0	
Berkeley parser (restricted)	87.3	31.0	86.6	29.0
Reranking parser (Charniak and Johnson, 2005)	92.0		91.4	

Table 4: Full treebank testing results showing labelled F1 and exact match accuracy for sentences of up to 40 words, and for all sentences. The results of several treebank parsers are also shown (as reported in the literature, hence the missing values), representing a baseline (PCFG), systems similar to our own (DOP, Berkeley) and state-of-the-art (Berkeley, Reranking parser). Berkeley (restricted) uses simplified data preprocessing as compared to Berkeley; the simplified preprocessing is the same as used in our system, so provides a more fair comparison.

threshold, which impedes the model's ability to learn highly lexicalized fragments. The grammar sizes are not strictly comparable, because we are comparing different types of grammar. For our TSG models we report the number of CFG productions in the transformed MAP PCFG, in which non-zero count TSG rules typically rewrite as many CFG rules[17] and CFG rules from the base distribution are replicated up to four times. Nevertheless the trend is clear: our model produces similar results to a state-of-the-art parser, and does so using a similar sized grammar. With additional rounds of split-merge training the Berkeley grammar grows exponentially larger (200K rules after six iterations). Our TSG grammar is also far smaller than the full DOP grammar induced from this data set, which extracts every possible TSG rule from the training set with no size limit, and has approximately 700K rules.

6.2 Full Treebank

We now train the model on the full training partition of the Penn treebank, using sections 2–21 (see Table 2 for corpus statistics). We initialise the sampler using a converged model from the end of a sampling run on the small data set and run the blocked Metropolis Hastings sampler for 20,000 iterations. The MAP PCFG approximation had 156k productions and training took 1.2 million seconds in total or 61 seconds per iteration.[18] We repeated this three times and present the averaged results in Table 4.

17. The encoding of TSG rules could be made more compact by skipping the internal rewrite steps, instead directly rewriting the transformed root node as the rule's frontier. This would mean that each input TSG rule would produce only two rules in the transformed CFG. It would also affect the choice of parsing algorithm because the transformed grammar would no longer be binary.
18. Measured using a single core of an AMD Opteron 2.6GHz machine.

The TSG models far surpass the MAP PCFG baseline, while the relative orderings of the different parsing algorithms corroborate our earlier evaluation on the small training set. The model outperforms the most comparable DOP result, although the numbers are not strictly comparable as Zuidema (2007) used an enriched nonterminal set for testing. However, our results are still well below state-of-the art parsers, and even underperform the Berkeley parser when it is restricted to the same preprocessing steps for rare tokens and binarization as we used (results labelled *restricted* in Table 4). But we must bear in mind that these parsers have benefited from years of tuning to the Penn-treebank, where our model is much simpler and is largely untuned. We anticipate that careful data preparation, model tuning and improved inference algorithms would greatly improve our model's performance, bringing it closer to the state-of-the-art.

6.3 Sampling Strategy

Next we investigate which of the two sampling methods is more effective for training the model. Recall that in Section 4 we described a blocked sampler and a local sampler; these samplers differ in the number of variables that they update in every sampling step. We return to the small training sample for these experiments. Figure 9 shows how the log posterior over the training set varies with the sampling iteration and with time for the two different sampling algorithms. It is clear that the blocked MH sampler exhibits faster convergence in general than the local Gibbs sampler, despite being somewhat slower per iteration. The speed difference is fairly minor, amounting to roughly a 50% increase in time over the local sampler,[19] although on the full data set the time differential reduces to 19%. This difference is largely due to the cost of performing the grammar transformation, which could potentially be further optimised to reduce the gap.

Figure 9 shows the results for a number of different initialisations, using *minimal* elementary trees where every node is a substitution point (the CFG analysis), initialising the substitution variables uniformly at random (*even*), and using *maximal* elementary trees where no nodes are substitution points. The blocked sampler is more robust to starting point than the local sampler and converges faster in terms of iterations and total time in general. Interestingly, the blocked sampler converges faster with the maximal initialisation, which is due to the initialisation condition resulting in much smaller initial table counts, and therefore it is quite easy for the model to move away from that solution. However, with the minimal initialisation the counts begin with very high values, and therefore deviating from the initial solution will be much less likely. In contrast, the local sampler behaves in the opposite way with respect to initialisation, such that with the minimal initialisation it performs at or above the level of the blocked sampler. This is a surprising finding which contradicts our intuition about the mixing properties of the sampler and warrants further research.

In the above we have been comparing the log training posterior as a measure of the quality of the sampler, however it is not a given that a probable model is one with good parsing accuracy. Figure 10 shows that the posterior is highly correlated with generalisation F1, with a Pearson's correlation efficient of 0.95 on this data, and therefore improving the sampler convergence will have immediate positive effects on performance. This is corroborated in Table 5, which shows the F1 scores using the final sample for each initialisation condition. The blocked sampler out-performs the local sampler for all initialisation conditions and has lower lower variance. Moreover, the blocked sampler is less dependent on its initialisation, performing well independent of initialisation. In

19. To account for the speed differential, the local samplers were run for 15k iterations and the blocked samplers for 10k iterations to produce Figure 9 and Table 5.

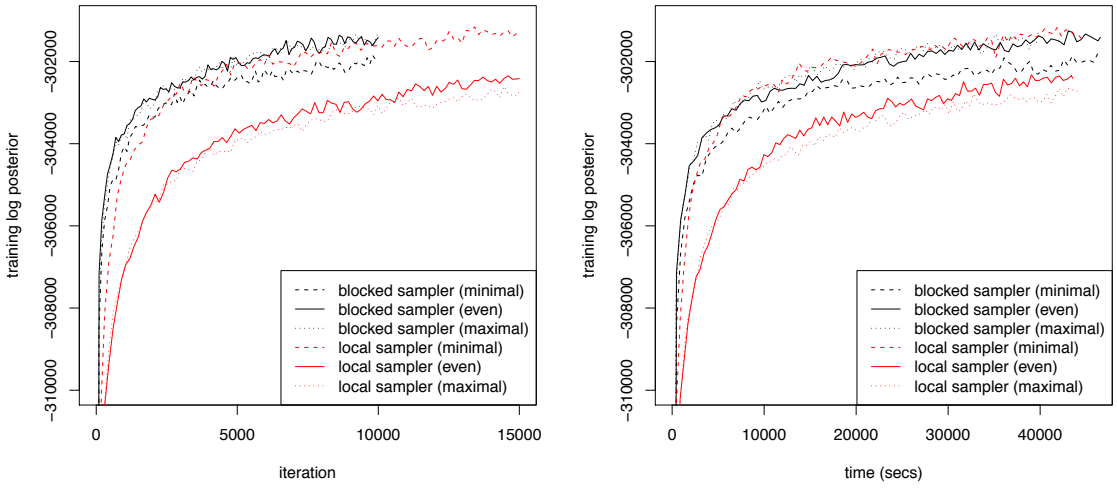


Figure 9: Training likelihood vs. iteration (left) or elapsed time (right). Both sampling methods were initialised in three different ways, *minimal* (all substitution variables set to 1, $\mathbf{x} = 1$), *even* ($x_d \sim \text{Uniform}(0,1)$) and *maximal* ($\mathbf{x} = 0$).

Sampling method	Initialisation	F1	σ_{F1}	Training time (s)
Local	minimal	78.2	0.50	44674
Local	even	78.3	0.31	43543
Local	maximal	78.5	0.51	44453
Block	minimal	78.5	0.18	46483
Block	even	78.6	0.35	46535
Block	maximal	78.6	0.38	39789

Table 5: Parsing performance and training time for the local versus blocked samplers with different initialisations. Results are shown on the development set using the MER parsing, reporting the mean F1 and standard deviation (σ_{F1}) from five independent runs. The blocked samplers were run for 10k iterations and the local sampler for 15k iterations iterations in order to allow all methods approximately the same running time.

contrast, the local sampler performs well only with the maximal initialisation, with which it roughly equals the blocked sampler in terms of F1 and log posterior (see Figure 9).

6.4 Discussion

The previous sections show that our model performs far better than a standard PCFG trained on the same corpus; it is natural to ask what kinds of rules it is learning that allow it to do so well. Figure 11 shows the grammar statistics for a TSG model trained on the full treebank training set. This model has a total of 72955 rules with an aggregate count of 733755. Of these, only 46% are

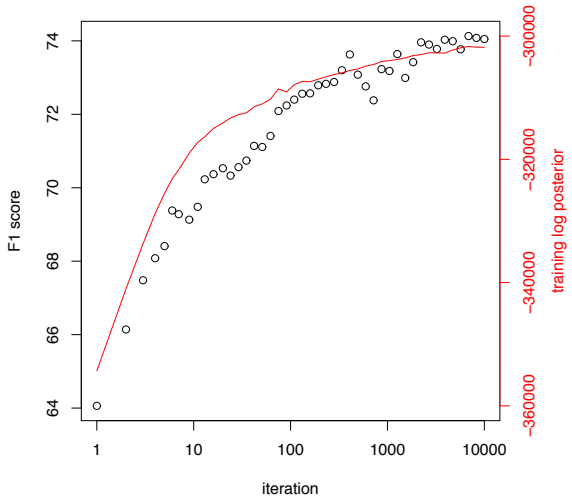


Figure 10: Likelihood and generalisation F1 are highly correlated. The black circles show the development F1 score (left axis) and the red line shows the training log-likelihood (right axis) during a Gibbs sampling run. The parsing results were obtained using Viterbi parsing with the MAP approximation grammar.

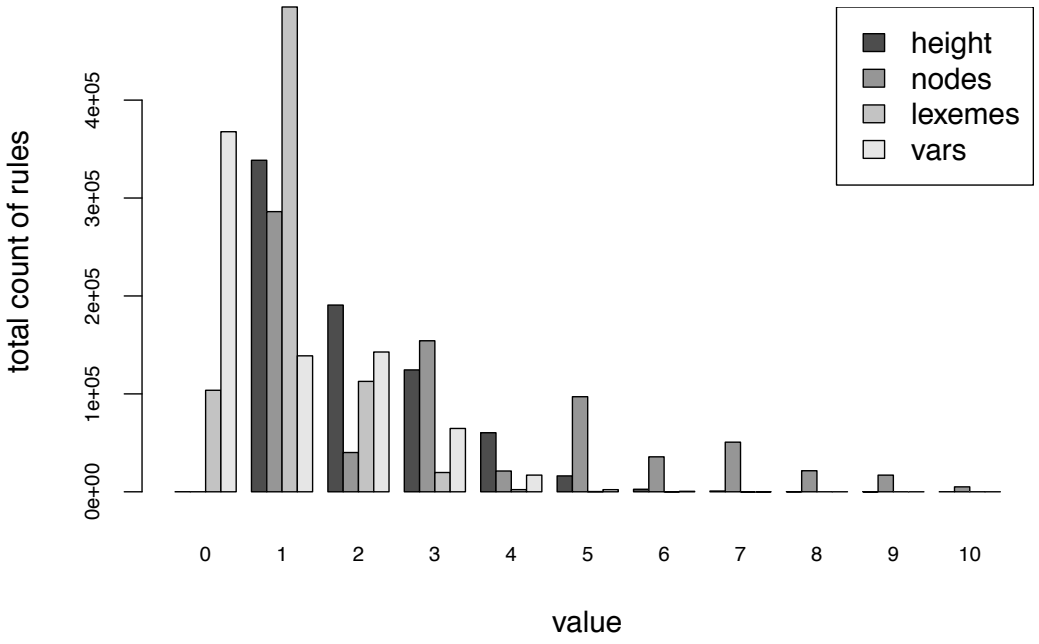


Figure 11: Grammar statistics for a TSG model trained on the full Penn treebank training set, showing the aggregate count for elementary trees of given height, number of nodes, terminals (lexemes) and frontier nonterminals (vars). An insignificant fraction of the rules had a height or number of nodes > 10; these have been truncated for display purposes.

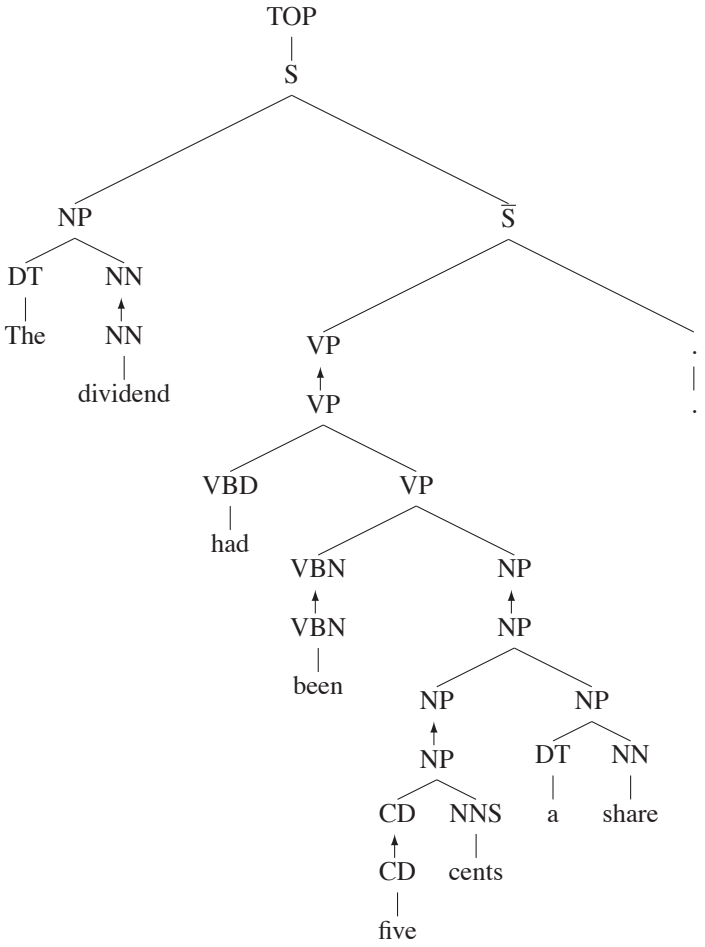


Figure 12: Inferred TSG structure for one of the training trees. Nonterminals shown with an overbar (e.g., $\overline{\text{S}}$) denote a binarized sub-span of the given phrase type.

CFG rules. The TSG rules vary in height from one to nineteen with the majority between one and four. Most rules combine a small amount of lexicalisation and a variable or two. This confirms that the model is learning local structures to encode, for example, multi-word units, subcategorisation frames and lexical agreement. The few very large rules specify full parses for sentences which were repeated in the training corpus. These complete trees are also evident in the long tail of node counts (up to 30; not shown in the figure) and counts for highly lexicalized rules (up to 11).

It is also instructive to inspect the inferred segmentations for individual trees in the training set. Figure 12 shows an example taken from the training set showing the learned derivation. Notice that the model has learned to use a large rule from the TOP node to capture the typical NP VP . sentence structure while lexicalizing the initial determiner, which is necessary to properly describe its capitalisation. It has also learnt a subcategorisation frame for `had` which specifies a VBN argument and an NP, and learnt the `<num> cents` and `NP a share` fragments, which are both common in the training corpus (Wall Street Journal articles).

To provide a better picture of the types of rules being learnt, Table 6 shows the top fifteen rules for three phrasal categories for the model trained on the full Penn treebank. We can see that many of these rules are larger than CFG rules, confirming that the CFG rules alone are inadequate to model the treebank. Two of the NP rules encode the prevalence of prepositional phrases headed by 'of' within a noun phrase, as opposed to other prepositions. Highly specific tokens are also incorporated into lexicalized rules.

The model also learns to use large rules to describe the majority of root node expansions (we add a distinguished TOP node to all trees). These rules mostly describe cases when the S category is used for a full sentence and includes punctuation such as the full stop and quotation marks. In contrast, the majority of expansions for the S category do not include any punctuation. The model has learnt to distinguish between the two different classes of S—full sentence versus internal clause—due to their different expansions.

Many of the verb phrase expansions have been lexicalized, encoding verb subcategorisation, as shown in Table 7. Notice that each verb here accepts only one or a small set of argument frames, indicating that by lexicalizing the verb in the VP expansion the model can find a less ambiguous and more parsimonious grammar. The lexicalized noun phrase expansions in Table 7 also show some interesting patterns, such as reconstituting binarized productions and lexicalizing prepositional phrase arguments. Of particular interest are the rules NP $\rightarrow$ CD (NN %) and NP $\rightarrow$ (NNP Mr.) NNP, in which the lexicalized item has a common tag but a very specific function. These rules are considerably more expressive than their CFG counterparts. Overall these results demonstrate that the model uses deep elementary trees to describe regular patterns in the data, thereby out-performing a simple PCFG. Despite having an infinite space of grammar productions, the induced grammars are compact, with the majority of probability mass on a small number of individually small productions.

7. Unsupervised Dependency Induction Experiments

Having considered the application of our non-parametric TSG model to supervised parsing, in this section we present an empirical evaluation for our model on the task of unsupervised parsing. Our approach to the unsupervised task differs from the supervised one in several ways. Two differences result from the fact that parse trees are not observed during training: first, we cannot use the local Gibbs sampler presented above, leaving the blocked sampler as our only option. Second, we cannot directly estimate P_C from the training data, so we must extend the model with an extra level of hierarchy (as described in Section 3) in order to induce P_C. A final difference in our unsupervised experiments is that we now focus on dependency grammar induction rather than phrase-structure induction; as outlined in Section 2.1, dependency grammars are a more feasible formalism for unsupervised grammar learning. Before presenting our experimental results, we first describe how to represent dependency grammars using our Bayesian TSG model.

To date, the most successful framework for unsupervised dependency induction is the Dependency Model with Valence (DMV, Klein and Manning, 2004). This approach induces a dependency grammar using a simple generative model in which the strings are said to be generated by a top-down process which recursively generates child dependents from each parent head word. This model has been adapted and extended by a number of authors (e.g., Cohen and Smith, 2009; Headden III et al., 2009); these approaches currently represent the state-of-the-art for dependency induction. In this section, we present a method for dependency grammar induction that incorporates the basic intuitions of the DMV, but is also capable of modelling larger units than just single parent-child

NP →
(DT the) $\overline{\text{NP}}$
NNS
NP ($\overline{\text{NP}}$ (CC and) NP)
JJ NNS
NP (PP (IN of) NP)
NP PP
(DT the) NN
DT ($\overline{\text{NP}}$ JJ NN)
NN NNS
(DT the) NNS
JJ $\overline{\text{NP}}$
(NP DT NN) (PP (IN of) NP)

VP →
VBD $\overline{\text{VP}}$
VBZ NP
VBD NP
VBZ $\overline{\text{VP}}$
VBP NP
VBP $\overline{\text{VP}}$
MD (VP VB NP)
(VBD said) (SBAR (S NP VP))
MD (VP VB $\overline{\text{VP}}$)
VP ($\overline{\text{VP}}$ (CC and) VP)
(VBZ is) ADJP
(VBD said) (SBAR (S (NP (PRP it)) VP))

PP →
(IN in) NP
(IN for) NP
(IN on) NP
(IN with) NP
TO NP
(IN at) NP
(IN from) NP
(IN by) NP
(TO to) NP
(IN of) NP
IN (S (VP VBG NP))
IN NP

ADJP →
JJ
(JJ able) S
RB JJ
RB $\overline{\text{ADJP}}$
(RB very) JJ
JJ ($\overline{\text{ADJP}}$ CC JJ)
ADJP ($\overline{\text{ADJP}}$ CC ADJP)
(RBR more) JJ
JJ PP
(NP CD (NNS years)) (JJ old)
(JJ UNK-LC)
(RB too) JJ

ADVP →
(RB also)
(RB now)
RB
(RB still)
(NP (DT a) (NN year)) (RBR earlier)
(RB already)
(RB UNK-LC-ly)
(RB again)
(RB just)
(RB then)
(RB never)
(RB currently)

TOP →
(S NP ($\overline{\text{S}}$ VP (. .)))
SINV
(S (NP (DT The) $\overline{\text{NP}}$) ($\overline{\text{S}}$ VP (. .)))
(S S ($\overline{\text{S}}$, $\overline{\text{S}}$))
(S (CC But) $\overline{\text{S}}$)
(S (PP (IN In) NP) ($\overline{\text{S}}$ (, ,) $\overline{\text{S}}$))
(S (NP (DT The) NN) ($\overline{\text{S}}$ VP (. .)))
(S (“ “) ($\overline{\text{S}}$ S ($\overline{\text{S}}$ (, ,) $\overline{\text{S}}$)))
(S (NP NP ($\overline{\text{NP}}$, ($\overline{\text{NP}}$ NP ,))) ($\overline{\text{S}}$ VP (. .)))
(S PP ($\overline{\text{S}}$ (, ,) ($\overline{\text{S}}$ NP ($\overline{\text{S}}$ VP (. .)))))
(S (NP (PRP He)) ($\overline{\text{S}}$ VP .))
(S (CC And) $\overline{\text{S}}$)

S →
NP VP
(VP TO (VP VB NP))
(NP PRP) VP
(VP TO (VP VB $\overline{\text{VP}}$))
(VP TO (VP VB ($\overline{\text{VP}}$ NP PP)))
(VP TO (VP VB))
(VP TO (VP VB PP))
(VP VBG NP)
(VP VBG $\overline{\text{VP}}$)
(NP (PRP We)) VP
(NP (PRP it)) VP
(NP (PRP I)) VP

Table 6: Most frequent expansions for a selection of nonterminals. Counts were taken from the final sample of a model trained on the full Penn treebank.

NP →	VP →
(DT the) $\overline{\text{NP}}$	(VBD said) (SBAR (S NP VP))
NP ($\overline{\text{NP}}$ (CC and) NP)	VP ($\overline{\text{VP}}$ (CC and) VP)
NP (PP (IN of) NP)	(VBD said) (SBAR (S (NP (PRP it)) VP))
(DT the) NN	VBD ($\overline{\text{VP}}$ (NP CD (NN %)) $\overline{\text{VP}}$)
(DT the) NNS	VP ($\overline{\text{VP}}$, ($\overline{\text{VP}}$ (CC and) VP))
(NP DT NN) (PP (IN of) NP)	VP ($\overline{\text{VP}}$ (, ,) ($\overline{\text{VP}}$ (CC but) VP))
(DT a) $\overline{\text{NP}}$	(VBD said) (SBAR (S (NP (PRP he)) VP))
(PRP$ its) $\overline{\text{NP}}$	(MD will) (VP VB $\overline{\text{VP}}$)
NP ($\overline{\text{NP}}$ (, ,) (SBAR WHNP (S VP)))	(VBD said) (SBAR (S (NP (DT the) NN) VP))
NP ($\overline{\text{NP}}$, ($\overline{\text{NP}}$ (SBAR WHNP (S VP)) (, ,)))	(VBD agreed) S
CD (NN %)	(VBZ is) (VP (VBN expected) S)
NP ($\overline{\text{NP}}$ (, ,) NP)	(VBP say) (SBAR (S NP VP))
(NP NNP (POS 's)) $\overline{\text{NP}}$	MD ($\overline{\text{VP}}$ (RB n't) (VP VB $\overline{\text{VP}}$))
(NNP Mr.) NNP	(VBZ says) (SBAR (S NP VP))
(PRP it)	(VBP do) ($\overline{\text{VP}}$ (RB n't) (VP VB NP))
(NP DT ($\overline{\text{NP}}$ JJ NN)) (PP (IN of) NP)	(MD will) (VP VB NP)
(DT a) ($\overline{\text{NP}}$ JJ NN)	(VBZ plans) S
NP (SBAR (WHNP (WDT that)) (S VP))	(VBD was) (VP VBN (PP (IN by) NP))
NP ($\overline{\text{NP}}$ (, ,) ($\overline{\text{NP}}$ NP (, ,)))	(VBD did) ($\overline{\text{VP}}$ (RB n't) (VP VB NP))
(NP (DT the) NN) (PP (IN of) NP)	VP ($\overline{\text{VP}}$ (CC but) VP)

Table 7: Most frequent lexicalized expansions for noun and verb phrases. Forms of `to be` and `to have` dominate the VP expansions and consequently have been excluded.

dependency relations. We approach the problem by representing dependency structures using the formalism of a CFG, and then applying our model (Section 3) to learn a TSG based on that CFG.

A dependency grammar can be represented in a CFG by labelling constituents with their head word, and encoding each dependency edge between a head and a child word in the grammar productions. This grammar has productions of the form $S \rightarrow H$ (word H heads the sentence), $H \rightarrow C\,H$ (C is a left child of H) and $H \rightarrow H\,C$ (C is a right child of H), where S is the start nonterminal and the H and C denote head and child nonterminals, respectively, which are both drawn from the same alphabet as the terminals (in our case part-of-speech tags). Unfortunately parsing with this representation is inefficient, having an asymptotic time complexity of $O(|\mathbf{w}|^5)$.[20] The complexity can be improved to $O(|\mathbf{w}|^3)$ by replicating (splitting) each terminal and processing all left and right dependents of each head word separately (Eisner, 2000). This is illustrated in Figure 13 where the leaves are all replicated with the l and r subscripts, while the spans defined by the tree structure denote the left and right dependents of each head word. Here we employ the *fold-unfold* representation (Johnson, 2007) that generalises Eisner's (2000) split-head parsing algorithm by defining an equivalent CFG under which standard inference methods can be used. Table 8 shows the CFG grammar for the DMV model (CFG-DMV), while Figure 13 shows the derivation in this grammar for the example sentence in Figure 2. The key insight to understanding the nonterminals in this grammar is that the subscripts encode the terminals at the boundaries of the span dominated by that nonterminal. For example the nonterminal L_H encodes that the right most terminal spanned by this constituent is H

20. This is a result of the usual CYK complexity of $O(G^2|\mathbf{w}|^3)$, and the grammar constant G being equal to the number of terminals $|\mathbf{w}|$ in the sentence.

(and the reverse for ${}_H\mathrm{R}$), while ${}_A\mathrm{M}_B$ encodes that A and B are the left and right terminals of the span. The superscripts * and ^{1} denote the valency of the head: both indicate that the head has at least one attached dependent in the specified direction, with ^{1} indicating that the head will continue to attach more children. The time complexity of inference in this grammar is only $O(|\mathbf{w}|^3)$ because each span in the parse chart bounded by terminals A and B can only contain nonterminal labels from the set $\{\mathrm{L}_B, \mathrm{L}^*_B, \mathrm{L}^1_B, {}_A\mathrm{R}, {}_A\mathrm{R}^*, {}_A\mathrm{R}^1, {}_A\mathrm{M}_{B^*}, {}_{A^*}\mathrm{M}_B, \mathrm{S}\}$. Consequently the grammar constant is fixed rather than quadratic in the sentence length.

We apply our TSG model to unsupervised dependency grammar induction using the CFG-DMV as the underlying grammar representation. Our model is capable of learning tree fragments which combine multiple adjacent CFG productions, affording considerable additional modelling power above that of a PCFG. Thereby the model can learn to condition dependency links on the valence. For example by combining $\mathrm{L}_{NN} \rightarrow \mathrm{L}^1_{NN}$ and $\mathrm{L}^1_{NN} \rightarrow \mathrm{L}_{DT\ DT}\mathrm{M}_{NN^*}$ rules into an elementary tree the model can describe that the leftmost child of a noun (NN) is a determiner (DT). Moreover, the model can learn groups of dependencies that occur together by conjoining multiple L^1_H or ${}_H\mathrm{R}^1$ nonterminals. This can represent, for example, the complete preferred argument frame of a verb.

7.1 Experiments

We perform inference for the TSG-DMV model by drawing 1000 samples using the blocked Metropolis-Hastings sampler described in Section 4.2 and evaluate the model using the final sample. Given that we do not observe parse trees in training, we cannot use the local Gibbs sampler as it only allows the sampling of the segmentation of a fixed tree, not the tree itself. In order to parse sentences in the test set we use the Viterbi algorithm to find the maximum probability parse under the MAP grammar (see Section 5). All hyperparameters, $\mathbf{a}, \mathbf{b}$ and $\mathbf{s}$, are sampled after every ten full samples over the training set.

A final and important consideration is the initialisation of the sampler. Klein and Manning (2004) emphasised the importance of the initialiser for achieving good performance with their model. We employ Klein and Manning's *harmonic* initialiser which defines a PCFG in which all words have the same valency distribution and probability of being the sentence head, while the probability of a head word attaching to a child word is inversely proportional to the average distance between these words in the training corpus. To obtain the initial derivations for the sampler we take the Viterbi derivations under this PCFG.

We follow the standard evaluation regime for DMV style models by performing experiments on the text of the WSJ section of the Penn. Treebank (Marcus et al., 1993). The corpus statistics are reported in Table 9. Like previous work we pre-process the training and test data to remove the words and punctuation, training our models on the gold-standard part-of-speech tags.

It is difficult for an unsupervised model to learn from long training sentences as their structure is highly ambiguous, and therefore the majority of DMV based approaches have been trained on sentences restricted in length to ≤ 10 tokens. This has the added benefit of decreasing the runtime for experiments. We present experiments with this training scenario, plus an additional experiment where we increase the length cutoff to ≤ 15. For the ≤ 15 experiment we start by sampling only sentences up to length 10, then gradually relax this length cutoff until we are sampling all sentences up to length 15 after 900 samples.[21] The training data is taken from sections 2-21, while section 23 is used for evaluation (see Table 9). An advantage of our sampling based approach over previous work

21. This training method is similar in spirit to the Baby Steps algorithm (Spitkovsky et al., 2010).

CFG Rule	DMV Distribution	Description
$S \rightarrow L_H\ {}_H R$	$p(root = H)$	The head of the sentence is H.
$L_H \rightarrow H_l$	$p(STOP \mid dir = L, head = H, val = 0)$	H has no left children.
$L_H \rightarrow L_H^1$	$p(CONT \mid dir = L, head = H, val = 0)$	H has at least one left child.
$L_H^* \rightarrow H_l$	$p(STOP \mid dir = L, head = H, val = 1)$	H has no more left children.
$L_H^* \rightarrow L_H^1$	$p(CONT \mid dir = L, head = H, val = 1)$	H has another left child.
${}_H R \rightarrow H_r$	$p(STOP \mid dir = R, head = H, val = 0)$	H has no right children.
${}_H R \rightarrow {}_H R^1$	$p(CONT \mid dir = R, head = H, val = 0)$	H has at least one right child.
${}_H R^* \rightarrow H_r$	$p(STOP \mid dir = R, head = H, val = 1)$	H has no more right children.
${}_H R^* \rightarrow {}_H R^1$	$p(CONT \mid dir = R, head = H, val = 1)$	H has another right child.
$L_H^1 \rightarrow L_C\ {}_C M_{H^*}$	$p(C \mid dir = L, head = H)$	C is a left child of H.
${}_H R^1 \rightarrow {}_{H^*} M_C\ {}_C R$	$p(C \mid dir = R, head = H)$	C is a right child of H.
${}_C M_{H^*} \rightarrow {}_C R\ L_H^*$	$p = 1$	Structural rule.
${}_{H^*} M_C \rightarrow {}_H R^*\ L_C$	$p = 1$	Structural rule.

Table 8: The CFG-DMV grammar schema. Note that the actual CFG is created by instantiating these templates with part-of-speech tags observed in the data for the variables H and C. Valency (*val*) can take the value 0 (no attachment in direction *dir*) and 1 (one or more attachment). L and R indicates child dependents left or right of the parent; superscripts encode the stopping and valency distributions, X^1 indicates that the head will continue to attach more children and X^* that it has already attached a child.

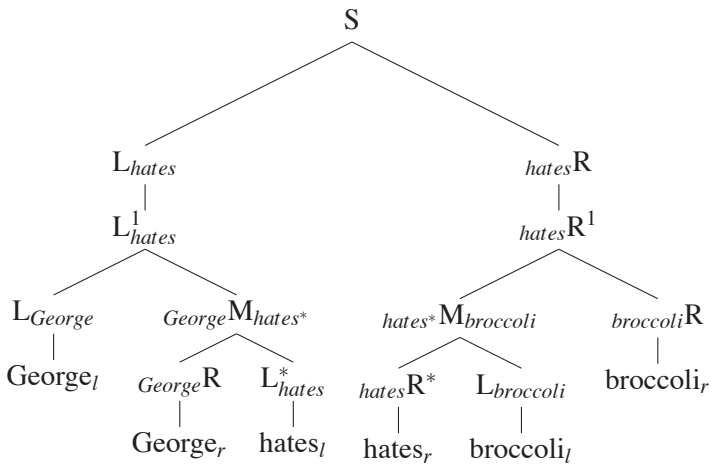


Figure 13: The CFG-DMV derivation for the example sentence *George hates broccoli*. The dependency parse for this sentence is given in Figure 2.

Partition	Sections	Words	Sentences
training$_{\leq 10}$	2–21	42505	6007
training$_{\leq 15}$	2–21	132599	12880
development$_{\leq 10}$	22	1805	258
test$_{\leq 10}$	23	2649	398
test$_{\leq 15}$	23	16591	1286
test$_{\leq \infty}$	23	49368	2416

Table 9: Corpus statistics for the training and testing data for the TSG-DMV model. All models are trained on the gold standard part-of-speech tags after removing punctuation.

is that we infer all the hyperparameters; consequently there is no need to tune on the development set (section 22).

The models are evaluated in terms of head attachment accuracy (the percentage of correctly predicted head dependency links for each token in the test data), on three subsets of the testing data. Although unsupervised models are better learnt from a corpus of short rather than long sentences, they must still be able to parse long sentences. The most commonly employed test set mirrors the training data by only including sentences ≤ 10, however we also include results for sentences ≤ 20 and the whole test set with no length restriction. As we are using MCMC sampling the result of any single run is non-deterministic and will exhibit a degree of variance. Our reported results are the mean and standard deviation (σ) from 40 sampling runs.

7.2 Discussion

Table 10 shows the head attachment accuracy results for our TSG-DMV, along with those of several other competitive models. Our model performs very well in comparison to the others; in particular it achieves the highest reported accuracy on the full test set by a considerable margin. On the $|\mathbf{w}| \leq 10$ test set the TSG-DMV is second only to the L-EVG model of Headden III et al. (2009). The L-EVG model extends DMV by adding additional lexicalisation, valency conditioning, interpolated back-off smoothing and a random initialiser. In particular Headden III et al. show that the random initialiser is crucial for good performance, but their approach requires training 1000 models to select a single best model for evaluation and leads to considerable variance in test set performance. Our model exhibits considerably less variance than those induced using this random initialiser, suggesting that the combination of the harmonic initialiser and blocked MH sampling may be a more practical training regime. The recently-proposed Adaptor Grammar DMV model of Cohen et al. (2010) is similar in many ways to our TSG model, incorporating a Pitman Yor prior over units larger than CFG rules. As such it is surprising that our model performs significantly better than this model. We can identify a number of possible explanations for these results: the Adaptor Grammar model is trained using variational inference with the space of tree fragments truncated, while we employ a sampler which can nominally explore the full space of tree fragments; and the tree fragments in the Adaptor Grammar model must be complete subtrees (i.e., they don't contain variables), whereas our model can make use of arbitrary tree fragments. An interesting avenue for further research would be to extend the variational algorithm of Cohen et al. (2010) to our TSG

Model	Initialiser	Directed Attachment Accuracy % on Section 23		
		$\|\mathbf{w}\| \leq 10$	$\|\mathbf{w}\| \leq 20$	$\|\mathbf{w}\| \leq \infty$
Attach-Right	-	38.4	33.4	31.7
EM (Klein and Manning, 2004)	Harmonic	46.1	39.9	35.9
Dirichlet (Cohen et al., 2009)	Harmonic	46.1	40.6	36.9
LN (Cohen et al., 2009)	Harmonic	59.4	45.9	40.5
SLN, TIE V&N (Cohen and Smith, 2009)	Harmonic	61.3	47.4	41.4
DMV (Headden III et al., 2009)	Random	$55.7_{\sigma=8.0}$	-	-
DMV smoothed (Headden III et al., 2009)	Random	$61.2_{\sigma=1.2}$	-	-
EVG smoothed (Headden III et al., 2009)	Random	$65.0_{\sigma=5.7}$	-	-
L-EVG smoothed (Headden III et al., 2009)	Random	$\mathbf{68.8}_{\sigma=4.5}$	-	-
Less is More WSJ15 (Spitkovsky et al., 2010)	Harmonic	56.2	48.2	44.1
Leap Frog WSJ45 (Spitkovsky et al., 2010)	Harmonic	57.1	48.7	45.0
Adaptor Grammar (Cohen et al., 2010)	Harmonic	50.2	-	-
TSG-DMV	Harmonic	$65.9_{\sigma=2.4}$	$58.3_{\sigma=2.3}$	$53.1_{\sigma=2.4}$
TSG-DMV WSJ15	Harmonic	$66.4_{\sigma=1.7}$	$\mathbf{58.5}_{\sigma=1.7}$	$\mathbf{53.4}_{\sigma=1.8}$
Supervised MLE (Cohen and Smith, 2009)	-	84.5	74.9	68.8

Table 10: Head attachment accuracy for our two TSG-DMV models (highlighted), and many other high performing models.

Tag	Frequency	Accuracy	Tag	Frequency	Accuracy
NN	564	0.70	CC	62	0.77
NNP	366	0.67	VBG	48	0.71
NNS	302	0.74	POS	26	0.92
DT	292	0.81	MD	22	0.82
IN	292	0.59	JJR	20	0.60
JJ	266	0.67	PRP$	18	1.00
VBD	266	0.79	EX	12	1.00
CD	224	0.21	WP	12	0.17
RB	212	0.40	JJS	10	0.40
PRP	132	0.94	WDT	6	1.00
VBZ	118	0.88	RP	6	0.33
VBN	84	0.71	RBS	4	1.00
VBP	78	0.67	UH	4	0.50
TO	70	0.43			

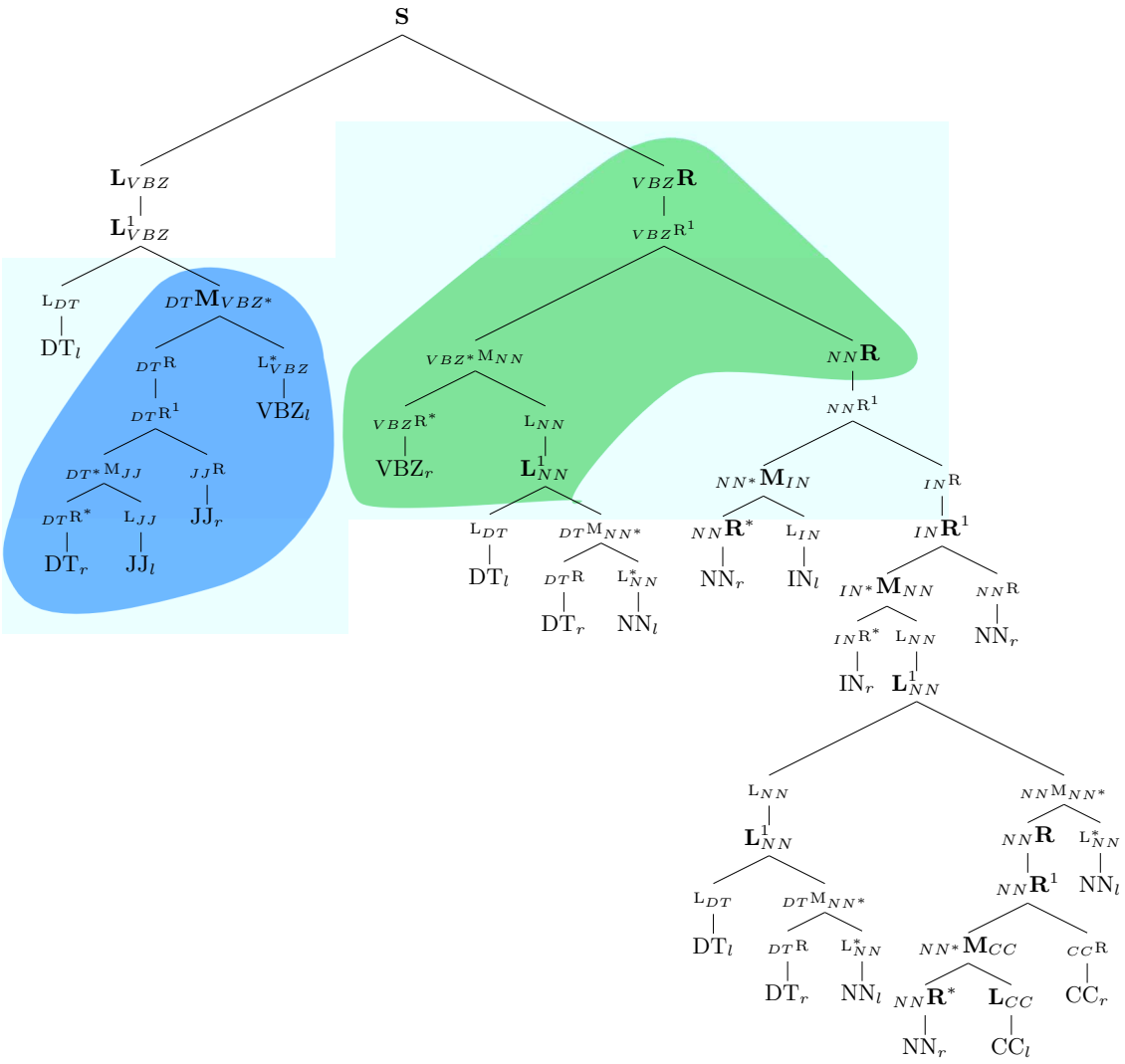
Table 11: Head attachment accuracy stratified by child tag, as measured on the held-out development set (WSJ 22, $|\mathbf{w}| \leq 10$). The tags are sorted by frequency.

model, possibly improving inference time while also allowing for the implementation to be more easily parallelised.

To illustrate the kinds of structures the model induces and the types of errors it makes, Figure 14 presents a representative example tree for a sentence from Section 22 of the WSJ. Though many of the elementary trees in this example resist an obvious linguistic explanation, on the right side of the derivation (highlighted in green) we see that the model has learnt to encode that the verb takes a single noun phrase as its object, while on the left (highlighted in blue) is a rule specifying the DT JJ subject dependent of the VBZ. This derivation is typical of the analyses produced by the model as it contains a number of dependency links which are inconsistent with the treebank. However we can see that the model has learnt to analyse the noun, verb and preposition phrases in a manner which is quite plausible, despite not matching the reference tree. In particular there would seem to be little obvious reason to prefer the treebank's analysis of the conjunction phrase ('either apathy or civility') over that produced by the unsupervised model. This highlights the difficulty in evaluating the output of unsupervised grammar induction algorithms against a treebank reference; in this instance it is clear that the analysis found by the model, despite its low accuracy, could be very useful for a downstream NLP application reliant on a dependency analysis.

For further analysis Tables 11 and 12 show the accuracy of the model at predicting the head for each tag type and the accuracy for dependencies spanning a given number of tokens. Clearly the model is far more accurate when predicting short dependencies, a result that is also reflected in the per-tag results. We also note that the model accurately predicts the head of the sentence 84% of the time, indicating an ability to capture the high level sentence structure. As mentioned above, conjunctions pose a particular difficulty with unsupervised models as the correct modelling of these remains a contentious linguistic issue. Nevertheless on conjunctions the model does achieve a reasonable level of agreement with the treebank.

(a) TSG-DMV representation. Large bold nodes indicate substitution points.



(b) Dependency tree representation. The red links below the sentence show where the treebank reference analysis differs from the predicted tree.

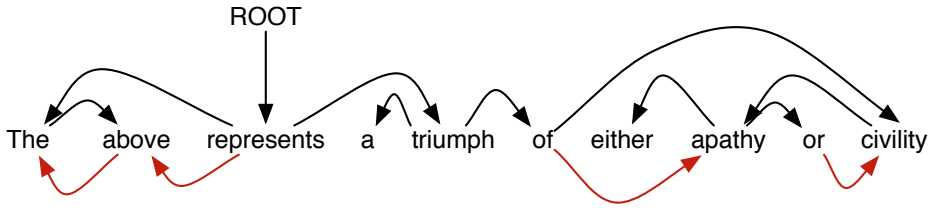


Figure 14: An example induced tree, shown as an elementary tree (a) and as a dependency tree (b). The sentence was taken from the development set: "The$_{DT}$ above$_{JJ}$ represents$_{VBZ}$ a$_{DT}$ triumph$_{NN}$ of$_{IN}$ either$_{DT}$ apathy$_{NN}$ or$_{CC}$ civility$_{NN}$".

Link Distance	Precision	Recall	F1-Score
ROOT	0.84	0.84	0.84
1	0.68	0.72	0.70
2	0.61	0.53	0.57
3	0.56	0.46	0.51
4	0.47	0.52	0.49
5	0.27	0.35	0.30
6	0.44	0.57	0.50
7	0.33	0.38	0.35
8	0.25	0.12	0.17

Table 12: Performance on dependency links of varying distance. Precision, recall and f-score on the WSJ Section 22 ($|\mathbf{w}| \leq 10$) held-out set.

TSG-DMV Rules			Frequency
L_{NN}	$\rightarrow$	$(L_{NN}\ (L^1_{NN}\ L_{DT}\ (_{DT}M_{NN^*\ DT}R\ L^*_{NN})))$	906
$_{IN}R$	$\rightarrow$	$(_{IN}R\ (_{IN}R^1\ _{IN^*}M_{NN\ NN}R))$	839
S	$\rightarrow$	$(S\ (L_{VBD}\ L^1_{VBD)}\ _{VBD}R)$	707
$_{JJ}M_{NN^*}$	$\rightarrow$	$(_{JJ}M_{NN^*\ JJ}R\ (L^*_{NN}\ NN_l))$	600
$_{NN^*}M_{NN}$	$\rightarrow$	$(_{NN^*}M_{NN\ NN}R^*\ (L_{NN}\ NN_l))$	589
L^1_{NN}	$\rightarrow$	$(L^1_{NN}\ L_{DT}\ (_{DT}M_{NN^*\ DT}R\ L^*_{NN}))$	587
L_{NNP}	$\rightarrow$	$(L_{NNP}\ (L^1_{NNP}\ (L_{NNP}\ NNP_l)\ _{NNP}M_{NNP^*}))$	540
L^*_{NN}	$\rightarrow$	$(L^*_{NN}\ (L^1_{NN}\ L_{JJ\ JJ}M_{NN^*}))$	500
$_{TO^*}M_{VB}$	$\rightarrow$	$(_{TO^*}M_{VB}\ (_{TO}R^*\ TO_r)\ L_{VB})$	437
$_{NN}R$	$\rightarrow$	$(_{NN}R\ (_{NN}R^1\ _{NN^*}M_{NN}\ (_{NN}R\ NN_r)))$	412
$_{DT}M_{NNS^*}$	$\rightarrow$	$(_{DT}M_{NNS^*}\ (_{DT}R\ DT_r)\ L^*_{NNS})$	397
$_{IN}R$	$\rightarrow$	$(_{IN}R\ (_{IN}R^1\ _{IN^*}M_{NNS}\ (_{NNS}R\ NNS_r)))$	328
L_{NNS}	$\rightarrow$	$(L_{NNS}\ (L^1_{NNS}\ L_{DT\ DT}M_{NNS^*}))$	326
$_{IN}M_{CD^*}$	$\rightarrow$	$(_{IN}M_{CD^*}\ (_{IN}R\ IN_r)\ (L^*_{CD}\ CD_l))$	302
$_{NNS}M_{VBD^*}$	$\rightarrow$	$(_{NNS}M_{VBD^*}\ (_{NNS}R\ NNS_r)\ L^*_{VBD})$	296

Table 13: The fifteen most frequent TSG-DMV rules in the training data.

Table 13 lists the most frequent TSG rules learnt by the model. The most frequent rule at the top of the table models noun phrases, encoding the fact that determiners have no children and attach as a left child to a phrase headed by a noun. It is interesting to see that our model has used a TSG rule to analyse noun phrases in a way consistent with the treebank, whereas the original DMV model preferred the opposite analysis of having DTs as the heads of noun phrases (Klein and Manning, 2004). Both results could be supported from a linguistic standpoint (Abney, 1987), but nevertheless it is a notable outcome that our more powerful model prefers to head noun phrases with nouns. Further down the table we see another interesting rule: $_{TO^*}\mathrm{M}_{VB} \rightarrow (_{TO^*}\mathrm{M}_{VB}\ (_{TO}\mathrm{R}^*\ \mathrm{TO}_r)\ \mathrm{L}_{VB})$. This rule specifies that a verb phrase headed by an infinitive attaches as the first child of the particle *to* on its left. Here the model has used the tree fragment to encode that the verb must be the first right child of the particle, an analysis both consistent with the treebank and expressing a bias against any form of split infinitive construction.

8. Conclusion

In this work we have presented a non-parametric Bayesian model for inducing tree substitution grammars in both supervised and unsupervised settings. By incorporating a structured prior over elementary rules our model is able to reason over the infinite space of all such rules, producing compact and simple grammars. In doing so, our model learns local structures for latent linguistic phenomena, such as verb subcategorisation and lexical agreement.

Our experimental results indicate that our model holds significant potential for a range of grammar induction tasks. In experiments using a treebank for training, we showed that the induced TSG grammars strongly out-perform standard PCFGs, and are comparable to a state-of-the-art parser on small data samples. While our results when training on the full treebank are well shy of the best available parsers, we have proposed a number of improvements to the model and the parsing algorithm that could lead to state-of-the-art performance in the future. Our second set of experiments removed the reliance on a treebank and showed that our TSG model achieves performance similar to the best recent models on sentences up to length 10, and outperforms all other models on longer sentences. This result is particularly promising, since it demonstrates the possibility of successfully learning complex hierarchical models, beyond just CFGs, without supervision. We hope that our work will open the door to further research into inducing linguistically rich grammars, such as tree adjoining and categorial grammars, that have so far been considered too difficult to learn from raw strings.

References

Steven Paul Abney. *The English Noun Phrase in its Sentential Aspect*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1987.

David Aldous. Exchangeability and related topics. In *École d'été de probabilités de Saint-Flour, XIII—1983*, pages 1–198. Springer, Berlin, 1985.

Phil Blunsom and Trevor Cohn. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213, Boston, Massachusetts, October 2010.

Rens Bod. Using an annotated language corpus as a virtual stochastic grammar. In *11th National Conference on Artificial Intelligence*, pages 778–783, Washington D.C., USA, July 1993.

Rens Bod. The problem of computing the most probable tree in data-oriented parsing and stochastic tree grammars. In *Proceedings of the 7th conference on European chapter of the Association for Computational Linguistics*, pages 104–111, Dublin, Ireland, 1995.

Rens Bod. Combining semantic and syntactic structure for language modeling. In *Proceedings of the 6th International Conference on Spoken Language Processing*, pages 106–109, Beijing, China, 2000.

Rens Bod. An efficient implementation of a new DOP model. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary, April 2003.

Rens Bod. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872, Sydney, Australia, July 2006.

Rens Bod, Remko Scha, and Khalil Sima'an, editors. *Data-oriented parsing*. Center for the Study of Language and Information — Studies in Computational Linguistics. University of Chicago Press, 2003.

Glenn Carroll and Eugene Charniak. Two experiments on learning probabilistic dependency grammars from corpora. In *Proceedings of the AAAI Workshop on Statistically-Based Natural Language Processing Techniques*, San Jose, California, 1992.

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180, Ann Arbor, Michigan, June 2005.

David Chiang and Daniel M. Bikel. Recovering latent information in treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 183–189, Taipei, Taiwan, 2002.

Alexander Clark. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 workshop on Computational Natural Language Learning*, pages 1–8, Toulouse, France, 2001.

Shay B. Cohen and Noah A. Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82, 2009.

Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Lon Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 321–328. 2009.

Shay B. Cohen, David M. Blei, and Noah A. Smith. Variational inference for adaptor grammars. In *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 564–572, 2010.

Trevor Cohn and Phil Blunsom. Blocked inference in Bayesian tree substitution grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 225–230, Uppsala, Sweden, July 2010.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556, Boulder, Colorado, June 2009.

Jason Eisner. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, October 2000.

Thomas S. Ferguson. A bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1(2):209–230, 1973.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. The infinite tree. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 272–279, Prague, Czech Republic, June 2007.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia, July 2006.

Joshua Goodman. *Parsing Inside-Out*. PhD thesis, Harvard University, 1998.

Joshua Goodman. Efficient parsing of DOP with PCFG-reductions. In Bod et al. (2003), chapter 8.

William P. Headden III, Mark Johnson, and David McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado, June 2009.

Hemant Ishwaran and Lancelot F. James. Generalized weighted Chinese restaurant processes for species sampling mixture models. *Statistica Sinica*, 13:1211–1235, 2003.

Mark Johnson. The DOP estimation method is biased and inconsistent. *Computational Lingusitics*, 28(1):71–76, March 2002.

Mark Johnson. Transforming projective bilexical dependency grammars into efficiently-parsable cfgs with unfold-fold. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 168–175, Prague, Czech Republic, June 2007.

Mark Johnson. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of ACL-08: HLT*, pages 398–406, Columbus, Ohio, June 2008a.

Mark Johnson. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27, Columbus, Ohio, June 2008b.

Mark Johnson and Sharon Goldwater. Improving nonparameteric bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June 2009.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 139–146, Rochester, NY, April 2007a.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. 2007b.

Aravind Joshi. Tree adjoining grammars. In Ruslan Mikkov, editor, *The Oxford Handbook of Computational Linguistics*, pages 483–501. Oxford University Press, Oxford, England, 2003.

Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA, July 2002.

Dan Klein and Christopher D. Manning. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 478–485, 2004.

Karim Lari and Steve J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 688–697, Prague, Czech Republic, June 2007.

Percy Liang, Michael I. Jordan, and Dan Klein. Type-based mcmc. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 573–581, Los Angeles, California, June 2010.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Igor′ A. Mel′čuk. *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany, 1988.

Bernard Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20 (2):155–172, 1994.

Radford Neal. Slice sampling. *Annals of Statistics*, 31:705–767, 2003.

Timothy J. O'Donnell, Noah D. Goodman, and Joshua B. Tenenbaum. Fragment grammar: Exploring reuse in hierarchical generative processes. Technical Report MIT-CSAIL-TR-2009-013, MIT, 2009.

Slav Petrov. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Los Angeles, California, June 2010.

Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 404–411, Rochester, NY, April 2007.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July 2006.

Jim Pitman. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102:145–158, 1995.

Jim Pitman. *Combinatorial Stochastic Processes*. Springer-Verlag, New York, 2006.

Jim Pitman and Marc Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900, 1997.

Matt Post and Daniel Gildea. Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48, Suntec, Singapore, August 2009.

Detlef Prescher, Remko Scha, Khalil Sima'an, and Andreas Zollmann. On the statistical consistency of DOP estimators. In *Proceedings of the 14th Meeting of Computational Linguistics in the Netherlands*, Antwerp, Belgium, 2004.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, 2010.

Fei Xia. *Automatic grammar generation from two different perspectives*. PhD thesis, University of Pennsylvania, 2002.

Andreas Zollmann and Khalil Sima'an. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, 10(2):367–388, 2005.

Willem Zuidema. Parsimonious data-oriented parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 551–560, Prague, Czech Republic, June 2007.

Journal of Machine Learning Research 11 (2010) 3097-3135 Submitted 7/09; Revised 6/10; Published 11/10

Collective Inference for Extraction MRFs Coupled with Symmetric Clique Potentials

Rahul Gupta GRAHUL@CSE.IITB.AC.IN
Sunita Sarawagi SUNITA@IITB.AC.IN
Ajit A. Diwan AAD@CSE.IITB.AC.IN
Computer Science and Engineering
IIT Bombay, Powai
Mumbai, 400076, India

Editor: Ben Taskar

Abstract

Many structured information extraction tasks employ collective graphical models that capture inter-instance associativity by coupling them with various clique potentials. We propose tractable families of such potentials that are invariant under permutations of their arguments, and call them *symmetric clique potentials*. We present three families of symmetric potentials—MAX, SUM, and MAJORITY.

We propose cluster message passing for collective inference with symmetric clique potentials, and present message computation algorithms tailored to such potentials. Our first message computation algorithm, called α-pass, is sub-quadratic in the clique size, outputs exact messages for MAX, and computes $\frac{13}{15}$-approximate messages for Potts, a popular member of the SUM family. Empirically, it is upto two orders of magnitude faster than existing algorithms based on graph-cuts or belief propagation. Our second algorithm, based on Lagrangian relaxation, operates on MAJORITY potentials and provides close to exact solutions while being two orders of magnitude faster. We show that the cluster message passing framework is more principled, accurate and converges faster than competing approaches.

We extend our collective inference framework to exploit associativity of more general *intra-domain properties* of instance labelings, which opens up interesting applications in domain adaptation. Our approach leads to significant error reduction on unseen domains without incurring any overhead of model retraining.

Keywords: graphical models, collective inference, clique potentials, cluster graphs, message passing

1. Introduction

Markov Random Fields (MRFs) are the models of choice in various structured information extraction (IE) tasks such as part-of-speech tagging, NP-chunking, text segmentation, and named entity recognition. The goal of IE is to mark each token in a sentence with a label from a discrete set, like Person, Location, and Other. As illustrated in Figure 1(c), the basic MRF extraction model defines edge potentials between labels of adjacent words to capture first-order dependencies (Lafferty et al., 2001). The resulting chain model allows tractable exact inference that is, computing the maximum a-posteriori (MAP) labeling of the sentence is easy.

Apart from traditional settings, IE is now being increasingly used in many web scenarios, such as query-driven extraction (Gupta and Sarawagi, 2009; Carlson et al., 2010) and constructing on-the-fly relations from semi-structured web sources (Elmeleegy et al., 2009). However these setups are characterized by limited training data, which restricts the robustness of the resulting MRFs during deployment. One promising way to increase the robustness of MRFs during deployment is to strengthen the first-order dependencies of the model with extra long-range dependencies , resulting in more robust labelings. The goal of this paper is to exploit long-range dependencies in a principled manner.

There has been a lot of work on capturing long-range dependencies between labels of non-adjacent words. This typically includes dependencies of the form—if a token repeats in a document, then the labels of the repetitions should ideally be the same (Sutton and McCallum, 2004; Bunescu and Mooney, 2004; Krishnan and Manning, 2006; Finkel et al., 2005). In Figure 1(d), we show an example where the extra dotted edges join unigram recurrences. Such long-range dependencies are termed *associative* since they introduce positive couplings between labels of token pairs. Apart from textual tokens, associative dependencies have also been exploited while labeling documents (Lu and Getoor, 2003; Chakrabarti et al., 1998), image pixels (Taskar et al., 2004), and annotating Web documents (Kulkarni et al., 2009). Due to these couplings, MAP inference is done collectively across all sentences so as to maximize the sum of sentence-specific and long-range associative potentials. This joint MAP computation task is traditionally called **collective inference**.

Previous work on collective inference can be broadly classified into two categories. The first category defines a separate associative potential over every inter-instance edge (i.e., dotted edge in Figure 1(d)). Inference on such graphs is performed by viewing it as a graphical model with pairwise potentials. A variety of generic approximation algorithms have been used to solve this typically intractable inference task, including Loopy Belief propagation (Bunescu and Mooney, 2004; Sutton and McCallum, 2004), and Gibbs sampling (Finkel et al., 2005). The second category defines an associative potential for each clique that is created from all repetitions of a unigram and the potentials can take more general forms like the Majority function (Krishnan and Manning, 2006). Collective inference on this category of models is performed using local search algorithms such as Iterative Conditional Mode fitting (ICM) (Lu and Getoor, 2003; Chakrabarti et al., 1998) or two stage algorithms (Krishnan and Manning, 2006).

This paper unifies various collective extraction tasks with different forms of associative potentials under a single framework. We do this by employing a cluster graph representation of the collective model. Figure 1(e) illustrates the cluster graph for our toy example. The cluster graph comprises of one cluster per MRF-instance, and one cluster per clique with its corresponding associative potential. As in the traditional collective extraction models, we assume that a clique comprises of all the occurrences of a particular token. We emphasize that instead of clusters for chain models, the framework can also define clusters for any tractable component, such as a bipartite matching or an alignment, but since our interest is in IE tasks, we shall focus on chain models.

Collective inference in our model then simply corresponds to message passing in this cluster graph. This view of collective inference offers several advantages over earlier approaches. First, it allows us to plug in and study various clique potentials cleanly under the same cluster message passing umbrella. Second, it allows us to exploit special properties of the associative potentials to design combinatorial algorithms that are both more accurate and more efficient than existing algorithms. Specifically, we show that most associative potentials used in practice are **symmetric clique potentials**. Symmetric clique potentials are invariant under any permutation of their argu-

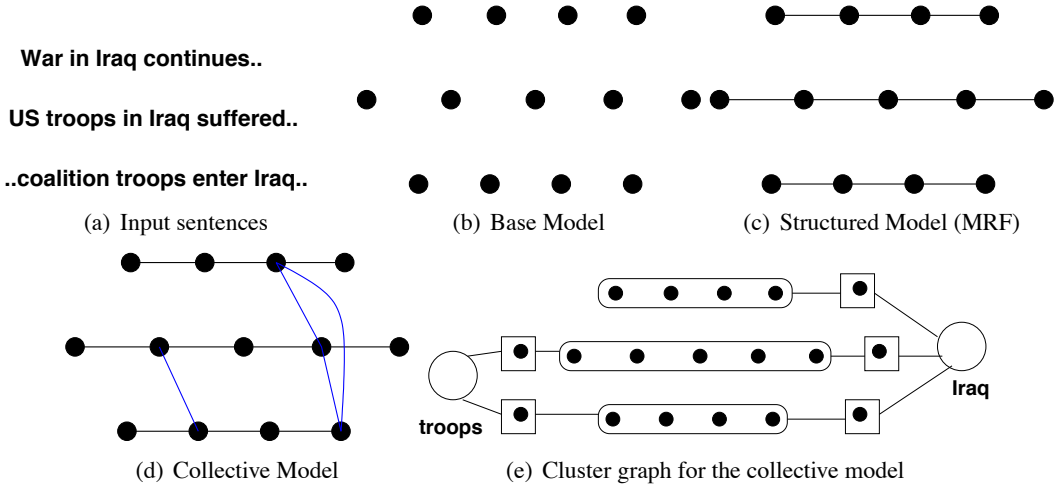
ments. So the value of a symmetric potential depends only on the counts of its distinct arguments, and not their position in the clique. For example in Figure 1(d), the clique potential on 'Iraq' would give a low score if its end vertices had different labels, regardless of which 'Iraq' vertex gets what label. We present three families of symmetric clique potentials that capture label associativity in different ways—MAX, SUM (which subsumes the popular Potts potential), and MAJORITY. The most crucial component of the collective inference problem is then to efficiently compute outgoing messages from clique clusters governed by symmetric potentials. We denote this sub-problem as **clique inference**.

1.1 Contributions

We show that the overall collective inference problem is intractable even for the case of two labels. We therefore concentrate on designing efficient and accurate message passing algorithms on our special cluster graph. We present a suite of efficient combinatorial algorithms tailored to specific symmetric clique potentials for the clique inference sub-problem. We devised a combinatorial algorithm called α-pass that computes exact outgoing messages for cliques with MAX potentials, and also for any arbitrary symmetric clique potential over two labels. We show that α-pass provides a $\frac{13}{15}$-approximation for clique inference with the well-known and NP-hard Potts potential with the SUM family. We show that this analysis is tight, and that the corresponding clique inference bound by alternative schemes like α-expansion, LP-rounding, TRW-S and ICM are either $\frac{1}{2}$. or at best locally optimal. Further, the runtime of α-pass is $O(mn\log n)$ where n is the clique size, and m is the number of labels. We also show that α-pass can be generalized to provide a better approximation of $\frac{8}{9}$, but with a runtime of $O(m^2 n\log n)$. Alternative clique inference approaches such as the graph-cut algorithm of Boykov et al. (2001) and the tree-reweighted message passing (TRWS) algorithm of Kolmogorov (2006) are quadratic in n. We present a new Lagrangian-relaxation based algorithm, called LR, for MAJORITY potentials. The LR algorithm is nearly exact in practice but is usually two orders of magnitude faster than an exact LP-based algorithm.

Our experiments show that computing a rich set of messages leads to significantly more accuracy gains over other collective inference approaches that do not compute such messages, for example, Krishnan and Manning (2006). We also show that decomposing the problem over the cluster graph and employing fast message computation algorithms helps our collective inference scheme converge one-order faster than alternatives like loopy belief propagation. In short, we show that it makes more sense to compute messages at a cluster level, and we provide fast and accurate algorithms to do so.

We then extend our collective inference framework to capture associativity of a more general kind than just labels of unigram repetitions. We encourage associativity of *properties* of labelings of records appearing as a group. We apply this framework to domain adaptation, where a model trained on one or more domains is deployed on a different but related domain. For example, a model trained for extracting fields of a bibliographic records is deployed on all records coming from the homepage of a single author. The bibliographic entries from the same author's publications homepage would predominantly have the same style, such as order of labels, say Title followed by Author(s) followed by Venue, or the HTML tag before a Title. While we do not know the property values across the records apriori, we do know that they will be largely unimodal across records within a given domain. We use this collective signal to couple together the labelings of intra-domain records, and show how our collective inference framework naturally extends to this scenario and provides significant reduction in error.



(a) Input sentences (b) Base Model (c) Structured Model (MRF)

(d) Collective Model (e) Cluster graph for the collective model

Figure 1: Various models for named-entity recognition illustrated on a small corpus. Figure 1(e) shows the cluster graph for the collective model of (d) with one cluster per sentence (shown as flat chains), and one cluster per associative clique (shown as big circles).

At this point, we note that our previous work (Gupta et al., 2007) focused primarily on the α-pass algorithm along with a sketch of some of its properties. This paper presents a detailed treatment of the algorithm and its generalized version, with rigorous proofs of their approximation bounds, including some new bounds for a variant of the clique inference objective with Potts potential. In addition, new material in this paper includes the LR algorithm for the MAJORITY potential, extension of our framework to include properties-based associativity, and empirical studies of various collective inference techniques.

We also note that subsequent to our work (Gupta et al., 2007), there has been an abundance of research on higher order clique potentials in last few years, primarily by the computer vision community. While their applications and basic graph structure are different (grids vs chains), many of their ideas are highly relevant—either in terms of special algorithms for clique/collective inference (Komodakis et al., 2007a; Komodakis and Paragios, 2009; Kumar and Torr, 2008a; Werner, 2008), reduction of clique potentials to pairwise potentials (Kohli et al., 2009; Ishikawa, 2009), and special families of clique potentials (Potetz and Lee, 2008; Rother et al., 2009; Tarlow et al., 2010). We will present theoretical and empirical comparisons and draw parallels with the relevant schemes later on in this paper.

1.2 Outline

In Section 2, we present the MRF model for extraction and define the collective inference problem in the traditional setup of unigrams. We show that even with this setup, collective inference remains NP-hard. In Section 3, we discuss the cluster message passing algorithm for collective inference, and introduce the *clique inference* sub-problem which formalizes the task of computing outbound messages from clique clusters. Then in Section 4, we describe the MAX, SUM, and MAJORITY families of symmetric potentials. In Section 5 we present the α-pass and LR clique inference algorithms,

and analyze their approximation quality. In Section 6, we extend our framework to capture more general associative properties. Section 7 contains experimental results of three types: (a) quality of the α-pass and LR algorithms; (b) effect of modeling more general associative properties in a domain adaptation task; and (c) collective inference via cluster message passing vs alternatives. Finally, Section 8 discusses prior art and Section 9 contains conclusions and a discussion of future work.

2. Collective Inference with MRFs

We now review the basic first-order MRF model for information extraction, and then formally define the collective inference problem over various coupled MRFs. We denote a sentence by bold $\mathbf{x}$, and its labeling vector by $\mathbf{y}$. Consequently, x_i and y_i denote the i^{th} token in $\mathbf{x}$ and its label respectively. The MRF defines a log-linear model parameterized by a weight vector $\mathbf{w}$, where the conditional probability of a labeling is given by:

$$\log P(\mathbf{y}|\mathbf{x},\mathbf{w}) = \sum_i \phi_i(y_i, y_{i-1}; \mathbf{x}, \mathbf{w}) - \log Z_{\mathbf{w}}.$$

Here $\phi_i(y_i, y_{i-1}; \mathbf{w})$ is the log of the potential on the edge between tokens i and $i-1$, These potentials are local, that is, they depend only on the labels of the edge and $\log Z_{\mathbf{w}}$ is the normalization factor. During inference, we compute the most probable labeling of $\mathbf{x}$:

$$\arg\max_{\mathbf{y}} \log P(\mathbf{y}|\mathbf{x},\mathbf{w}) = \arg\max_{\mathbf{y}} \sum_i \phi_i(y_i, y_{i-1}; \mathbf{x}, \mathbf{w}).$$

If there are m possible labels at each token, and n tokens in $\mathbf{x}$, then exact inference can be done using max-product message passing in $O(nm^2)$ time.

We now move to the collective model whose toy example is illustrated in Figure 1(d). Let there be N sentences $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$. Correspondingly, we have N conditional distributions $P(\mathbf{y}^k|\mathbf{x}^k, \mathbf{w})$, $k = 1, \ldots, N$. We use the shorthand $\phi_i^k(y_i^k, y_{i-1}^k)$ to refer to $\phi_i^k(y_i^k, y_{i-1}^k; \mathbf{x}^k, \mathbf{w})$.

Let t denote a token that repeats across the sentences, and let $\mathcal{T}$ denote the set of all such tokens. For a repeating token t, let $D(t)$ be the set of all (sentence-id, token-id) locations where it occurs in the N sentences, that is, $D(t) = \{(k,i)|x_i^k = t\}$. We express the conformance in the label assigned to positions in $D(t)$ with a higher order clique potential $\mathrm{C}_t(\{y_i^k\}_{(k,i)\in D(t)})$. Our collective MRF model that couples sentences using unigrams is then given by (up to a normalization constant):

$$\begin{aligned} \log P(\{\mathbf{y}^k\}_{k=1}^N | \{\mathbf{x}^k\}_{k=1}^N, \mathbf{w}) &\approx \sum_{k=1}^N \log P(\mathbf{y}^k|\mathbf{x}^k,\mathbf{w}) + \sum_{t\in\mathcal{T}} \mathrm{C}_t(\{y_i^k\}_{(k,i)\in D(t)}) \qquad (1) \\ &\approx \left(\sum_{k=1}^N \sum_{i=1}^{|\mathbf{x}^k|} \phi_i^k(y_i^k, y_{i-1}^k) \right) + \sum_{t\in\mathcal{T}} \mathrm{C}_t(\{y_i^k\}_{(k,i)\in D(t)}). \end{aligned}$$

The clique potential C_t has two important properties: First, it is invariant under any permutation of its arguments—that is, it is a *symmetric* function of its input. Thus, if there are m possible labels, we can represent the arguments with an m-bin histogram. For example, if $m = 3$ and input is $\{1,2,1,1,2,3,2\}$, then $\mathrm{C}_t(S)$ depends on the 3-bin histogram with values $3,3,1$. Second, C_t is *associative*—that is, it favors agreement of labels in its argument set, and it maximized when all arguments have the same label.

All collective extraction methods proposed in the literature fit the above framework. The earliest and the most popular collective model used associative potentials over every pair of occurrences of t (Sutton and McCallum, 2004; Finkel et al., 2005). This pairwise potential was a Potts potential—1 if the pair had matching labels and 0 otherwise. Clearly, we can define an appropriate C_t that mimics this behavior while staying symmetric. Consider the clique potential:

$$C_t(\{y_i^k|(k,i) \in D(t)\}) = \sum_{(k,i),\ (l,j)\in D(t)} \delta(y_i^k = y_j^l).$$

This is equivalent to all the pairwise potentials for t. Also, using the histogram view, this can be re-written up to a constant as $\sum_{\alpha=1}^{m} n_\alpha^2$, where n_α is the bin count for label α in the histogram corresponding to $\{y_i^k|(k,i) \in D(t)\}$. Other known collective models like the ones proposed by Lu and Getoor (2003) and Krishnan and Manning (2006) can also be cast using the 'Majority' symmetric clique potential as we shall see in later sections.

Having defined our collective model, and shown that it subsumes popular collective models, we are ready to define the collective inference objective.

Definition 1 (Collective Inference) *Collective inference is the task of finding labelings* $\mathbf{y}^1, \ldots, \mathbf{y}^N$ *that jointly maximize the probability of the collective model (Equation 1):*

$$\arg\max_{\mathbf{y}^1,\ldots,\mathbf{y}^N} \left(\sum_{k=1}^{N} \sum_{i=1}^{|\mathbf{x}^k|} \phi_i^k(y_i^k, y_{i-1}^k) \right) + \sum_{t\in\mathcal{T}} C_t(\{y_i^k\}_{(k,i)\in D(t)}). \tag{2}$$

In general terms, collective inference corresponds to labeling the nodes of a generic cyclic graph. However, our graph has a special structure—it is composed of chains and cliques, and although the intra-chain edge potentials ϕ can be arbitrary, the clique potentials are associative and symmetric. However we next prove that even in this setup, collective inference is NP-hard.

Theorem 2 *The collective inference problem is NP-hard even with just two labels.*

Proof We reduce the well-known NP-hard problem of finding MAX-CUTs in an arbitrary graph G to our collective inference problem (called CI). For each node $u \in G$ define a 2-node chain $u_1\ u_2$ in CI where each node can take binary labels and with edge potential $\phi^u(y,y') = M\delta(y \neq y')$. M is a large positive number $> 2E$, where E is the number of edges in G. For an edge $(u,v) \in G$, we add three cliques: $(u_1,v_1),(u_1,v_2)$ and (u_2,v_1). All three clique potentials are $C_t(\{y,y'\}) = \delta(y = y')$.

The key observation is that G has a cut of size $\geq k$ iff CI has a map score of $\geq Mn+E+k$ where n is the number of nodes in G.

Suppose G has a cut (A,B) of size $\geq k$. Define a labeling in CI where all variables in $A' \triangleq \{v_1|v \in A\} \cup \{v_2|v \in B\}$ are labeled 0 and the rest B' are labeled 1. Then for every $u \in G$, (u_1,u_2) are labeled differently, thus the total contribution from the edge potentials within chains of CI is Mn. For an edge $(u,v) \in G$ that is part of the cut (A,B) with $u \in A, v \in B$, two cliques (u_1,v_2) and (u_2,v_1) have both their arguments labeled the same whereas for all other edges only one of the cliques (u_1,v_1) reaches such conformance. Thus, the total contribution from clique potentials is $E+k$. The converse holds the same way. Since M is sufficiently large, the edge (u_1,u_2) in CI will always have both ends labeled differently. So given a labeling in CI, we can define a cut in G by the subset of vertices v for which v_1 is labeled 0 in CI.

■

At this point we note that our collective inference objective is related to various bodies of work on joint inference, primarily in computer vision tasks. For the sake of continuity we defer our discussion of all related work to Section 8 and instead present our framework for maximizing the collective inference objective.

3. Cluster Graph based Collective Inference Framework

Having established that optimizing the objective in Equation 2 is NP-hard, one natural choice for approximating it is ordinary pairwise belief propagation on the collective graphical model. This involves passing messages along edges inside the chains, as well as along the clique edges. However this approach is not suitable due to many reasons. First, some symmetric potentials like the Majority potential cannot be decomposed along the clique edges, so we cannot compute the corresponding messages. Second, the approach does not exploit the special nature of the clusters and the symmetric potentials. Third, this approach is not tenable if we wish to extend the framework to capture associativities of more general properties of the kind discussed in Section 6.

Hence we adopt message passing on a special cluster graph where every chain and clique corresponds to a cluster. The clique cluster for a unigram is adjacent to all the chains that contain that unigram, as shown via singleton separator vertices in Figure 1(e). This setup of message passing on the cluster graph allows us to exploit potential-specific algorithms at the cliques, and at the same time work with any arbitrary symmetric clique potentials. Cluster graphs have been used effectively for message passing elsewhere as well (Yedidia et al., 2003; Duchi et al., 2007).

Let $m_{k\to t}$ and $m_{t\to k}$ denote message vectors from chain k to an incident clique t and vice-versa. A chain is said to be incident on a clique if there exists a position j in chain k which matches term t, that is, $(k, j) \in D(t)$. We assume that $D(t)$ is created such that from any chain only one position belongs to it. We next discuss how these messages are computed.

3.1 Message from an Instance to a Clique

The message $m_{k\to t}(y)$ for a $(k, j) \in D(t)$ is given by:

$$m_{k\to t}(y) = \max_{\mathbf{y}^k : y^k_j = y} \left(\sum_{i=1}^{|\mathbf{x}^k|} \phi^k(y^k_i, y^k_{i-1}) + \sum_{t' \neq t \in \mathcal{T}, (k,j') \in D(t')} m_{t'\to k}(y^k_{j'}) \right).$$

To compute $m_{k\to t}(y)$, we need to absorb the incoming messages from other incident cliques $t' \neq t$, and do the maximization while freezing the label of position j in chain k to y for a $(k, j) \in D(t)$. A clique t' is incident to chain k via only a singleton vertex so we can easily absorb the message $m_{t'\to k}$ by including it in the node potential of this vertex. After absorption, $m_{k\to t}$ can be computed using the same inference algorithm applicable to the chain.

3.2 Message from a Clique to an Instance

The more interesting message computation is that for $m_{t\to k}$ given a $(k,j) \in D(t)$. Let the clique t have n vertices. Then the message computation can be written as:

$$m_{t\to k}(y) = \max_{(y_1,\ldots,y_k=y,\ldots,y_n)} \sum_{(k',j')\in D(t)\ k'\neq k} m_{k'\to t}(y_{k'}) + \mathrm{C}_t(\{y_1,\ldots,y_n\}). \tag{3}$$

The maximization in Equation 3 can be re-written as

$$-m_{k\to t}(y) + \max_{(y_1,\ldots,y_k=y,\ldots,y_n)} \left(\sum_{(k',j')\in D(t)} m_{k'\to t}(y_{k'}) + \mathrm{C}_t(\{y_1,\ldots,y_n\}) \right). \tag{4}$$

The maximization term is an instance of the general *clique inference problem* defined as:

Definition 3 (Clique Inference) *Given a clique over n vertices, with a symmetric clique potential $C(y_1,\ldots,y_n)$, and vertex potentials ψ_{jy} for all $j \le n$ and labels y. Compute a labeling of the clique vertices that maximizes:*

$$\max_{y_1,\ldots,y_n} \sum_{j=1}^{n} \psi_{jy_j} + C(y_1,\ldots,y_n). \tag{5}$$

Thus the second term in Equation 4 can be seen as clique inference by defining $\psi_{jy} \triangleq m_{j\to t}(y)$ and $\mathrm{C} \triangleq \mathrm{C}_t$. To compute $m_{t\to k}(y)$, we can solve the clique inference problem with the easily enforceable constraint $y_k = y$. From now on, we refer to outbound message computation at the cliques as clique inference.

4. Symmetric Clique Potentials

Having established cluster message passing as our collective inference paradigm, and clique inference as our message computation tool, we turn our attention towards various families of symmetric clique potentials. As seen in Section 2, these associative clique potentials depend only on the histogram of label counts $\{n_y | y = 1,\ldots,m\}$ over the clique, n_y being the number of clique vertices labeled y. Thus for ease of notation, we will denote the arguments of a symmetric potential C_t by either the vertex labels $\mathbf{y}_t = (y_1,\ldots,y_n)$ or by its corresponding count histogram $\mathbf{n}(\mathbf{y}_t) = (n_1,\ldots,n_m)$. Here we clarify that $\mathbf{n}$ is used to denote the entire count histogram, n_y to denote the count for label y, while n denotes the clique size. An associative symmetric clique potential is thus maximized when $n_y = n$ for some y, that is, one label is given to all the clique vertices.

We consider specific families of clique potentials, many of which are currently used in real-life tasks. In Section 5 we will look at various potential-specific exact and approximate clique inference algorithms that exploit the specific structure of the potential at a clique.

In particular, we consider the three types of symmetric clique potentials listed in Table 1. They differ in the manner in which they reward skew in the count histogram $\mathbf{n}$.

4.1 MAX Clique Potentials

These clique potentials are of the form:

$$\mathrm{C}(n_1,\ldots,n_m) = \max_y f_y(n_y).$$

Name	Form	Remarks
MAX	$\max_y f_y(n_y)$	f_y is a non-decreasing function.
SUM	$\sum_y f_y(n_y)$	f_y non-decreasing. Includes Potts = $\lambda \sum_y n_y^2$.
MAJORITY	$f_a(\mathbf{n})$, where $a = \text{argmax}_y n_y$	A popular form is $f_a(\mathbf{n}) = \sum_y w_{ay} n_y$.

Table 1: Three kinds of symmetric clique potentials considered in this paper. $\mathbf{n} = (n_1, \ldots, n_m)$ denotes the counts of various labels among the clique vertices.

for arbitrary non-decreasing functions f_y. When $f_y(n_y) \triangleq n_y$, we get the *makespan* clique potential which has roots in the job-scheduling literature. This potential depends only on the biggest label count and ignores the other labels. Truncated version of this potential have been recently used in computer vision (Kohli et al., 2009).

In Section 5.1, we present the α-pass algorithm that computes messages for MAX clique potentials exactly in $O(mn \log n)$ time. MAX potentials are tractable and relatively simpler potentials, but most importantly, they provide key insights to deal with the more complex SUM potentials.

4.2 SUM Clique Potentials

SUM clique potentials are of the form:

$$\mathrm{C}(n_1, \ldots, n_m) = \sum_y f_y(n_y).$$

This family of potentials includes functions that aggregate the histogram skew over bins, for example, the entropy potential where $f_y(n_y) \propto n_y \log n_y$. One very interesting member is the case when the well-known Potts model is applied homogeneously on all edges of a clique. Let $\lambda > 0$ be the pairwise reward of assigning the same label to two nodes of an edge. The summation of these terms over a clique is equivalent (up to a constant) to the clique potential:

$$\mathrm{C}^{\text{Potts}}(n_1, \ldots, n_m) = \lambda \sum_y n_y^2.$$

This corresponds to the gini entropy of the histogram. We will show that the clique inference problem is NP-hard with the above potential and provide a $\frac{13}{15}$-approximation in Section 5.

We note that the traditional usage of Potts is in a minimization setting, that is, edges are penalized by some cost γ if their end vertices are labeled differently. The corresponding cost version of Potts is then $\mathrm{C}^{\min}(y_1, \ldots, y_n; \gamma) \triangleq \gamma \sum_{j>i} \delta(y_i \neq y_j)$. It is easy to see that these two versions are related by:

$$\mathrm{C}^{\text{Potts}}(y_1, \ldots, y_n; \lambda) = \lambda n^2 - \mathrm{C}^{\min}(y_1, \ldots, y_n; 2\lambda).$$

We show that our algorithm provides a $\frac{3}{2}$ approximation for the minimization version of the problem.

4.3 MAJORITY Clique Potentials

A MAJORITY potential is defined as:

$$\mathrm{C}(n_1, \ldots, n_m) = f_a(\mathbf{n}),\ a = \text{argmax}_y\ n_y. \tag{6}$$

An important special subclass is the linear majority potentials defined as:

$$\mathrm{C}^{\mathrm{Maj}} = \sum_y w_{ay} n_y,\ a = \mathrm{argmax}_y\ n_y.$$

This potential has been used for a variety of tasks such as link-based classification of web-pages (Lu and Getoor, 2003) and named-entity extraction (Krishnan and Manning, 2006). The role of the parameters w_{ay} is to capture the co-existence of some label pairs in the same clique. Co-existence allows us to downplay 'strict associativity' viz. it permits a few cliques vertices to have similar but not necessarily the same labels. For example, consider a clique made from occurrences of the unigram 'America'. However, some occurrences of America correspond to Location, while others might correspond to an Organization, say Bank of America. Also, it is rare for a location name to be shared with a person name. This can be captured by allowing a higher value of $w_{\alpha y}$ for the (Location, Organization) pair than for the (Location, Person) pair.

Unlike Potts potential, MAJORITY potential cannot be represented using edge potentials. We will present an exact polynomial time algorithm and several efficient approximations in Section 5.3.

5. Algorithms for Clique Inference

We will use $F(y_1,\ldots,y_n)$ to denote the clique inference objective in Equation 5. As short-hand, we will denote $F(y_1,\ldots,y_n)$ by $F(\mathbf{y}) = \psi(\mathbf{y}) + \mathrm{C}(\mathbf{y})$, where $\psi(\mathbf{y})$ is the vertex score (i.e., node potential) of the clique labeling $\mathbf{y}$ and the second term is the clique score (i.e., clique potential). Wlog assume that all the vertex terms ψ_{iy} are positive. Otherwise a constant can be added to all of them and that will not affect the maximization. The MAP clique labeling will be denoted by $\mathbf{y}^*$, and $\hat{\mathbf{y}}$ will denote a possibly sub-optimal labeling.

We show that the clique inference is easy for *any* $\mathrm{C}()$ with just two labels and in Section 5.1 we present an exact algorithm called α-pass for this case. We show that the same algorithm generalizes to give an exact solution for MAX potentials. We address the SUM and MAJORITY clique potentials respectively in Sections 5.2 and 5.3. Finally, in Section 5.4 we show how to extend the clique inference algorithms to efficiently batch the computation of multiple max-marginals.

5.1 α-pass Algorithm

We begin with MAX potentials. Recall that a MAX potential is of the form $\mathrm{C}(\mathbf{n}(\mathbf{y})) = \max_y f_y(n_y)$. We propose an exact inference procedure called α-pass (Algorithm 1) for such potentials.

The α-pass algorithm guesses that the dominant label in $\mathbf{y}^*$ is α, with a count of k. Of course we do not know α or k so all (α,k) combinations are tried out. For each (α,k) combination, α-pass computes the best k vertices to assign the label α. These k vertices are obtained by sorting all the vertices according to the criteria $\psi_{.\alpha} - \max_{\beta\neq\alpha}\psi_{.\beta}$, and picking the top-$k$ vertices. Every remaining vertex u is labeled with the best non-α label as per the vertex score, that is, with $\arg\max_\beta \psi_{u\beta}$. Let $\hat{\mathbf{y}}^{\alpha k}$ denote the clique labeling thus obtained in the $(\alpha,k)^{th}$ combination. Trying out all (α,k) combinations, α-pass returns the $\hat{\mathbf{y}}^{\alpha k}$ whose score $F(\hat{\mathbf{y}}^{\alpha k})$ is the highest.

It is straightforward to see that α-pass runs in $O(mn\log n)$ time by incrementally computing $F(\hat{\mathbf{y}}^{\alpha k})$ from $F(\hat{\mathbf{y}}^{\alpha(k-1)})$, that is, for each α, we can sort the vertices just once for all $k = 1,\ldots,n$. For clique potentials that are decomposable over the edges, as in Potts, this runtime is much better than ordinary belief propagation, which would cost $O(m^2n^2)$.

We now look at properties of α-pass.

Input: Vertex scores ψ, Clique Potential C
Output: Labeling $\hat{\mathbf{y}}$
Best = $-\infty$;
foreach *label* $\alpha \in \{1,\ldots,m\}$ **do**
  Sort the vertices in descending order according to the metric $\psi_{.\alpha} - \max_{\beta\neq\alpha}\psi_{.\beta}$;
  foreach $k \in \{1,\ldots,n\}$ **do**
    Assign the first k sorted vertices the label α;
    Assign the remaining vertices their individual best non-α label;
    s $\leftarrow$ score of this labeling under $F()$;
    if $s >$ *Best* **then**
      Best $\leftarrow$ s;
      $\hat{\mathbf{y}} \leftarrow$ current labeling;
    end
  end
end
return $\hat{\mathbf{y}}$;

Algorithm 1: The α-pass algorithm

Claim 5.1 *Assignment* $\hat{\mathbf{y}}^{\alpha k}$ *has the maximum vertex score over all* $\mathbf{y}$ *where* k *vertices are assigned label* α*, that is,* $\psi(\hat{\mathbf{y}}^{\alpha k}) = max_{\mathbf{y}:n_\alpha(\mathbf{y})=k}\psi(\mathbf{y})$.

Claim 5.2 *For* MAX *potentials,* $C(\hat{\mathbf{y}}^{\alpha k}) \geq f_\alpha(k)$.

Proof Label α has a count of k and the MAX potential considers the maximum over all label counts, which is at least k. ■

Theorem 4 *The* α*-pass algorithm solves the clique inference problem exactly for* MAX *clique potentials in* $O(mn\log n)$ *time.*

Proof Let $\mathbf{y}^*$ be the true MAP and let $\beta = \text{argmax}_y f_y(n_y(\mathbf{y}^*))$, $\ell = n_\beta(\mathbf{y}^*)$. Let $\hat{\mathbf{y}}$ be the labeling returned by α-pass. We have:

$$\begin{aligned}
F(\hat{\mathbf{y}}) &= \max_{1\leq\alpha\leq m, 1\leq k\leq n} F(\hat{\mathbf{y}}^{\alpha k}) \\
&\geq F(\hat{\mathbf{y}}^{\beta\ell}) \\
&= \psi(\hat{\mathbf{y}}^{\beta\ell}) + \mathrm{C}(\hat{\mathbf{y}}^{\beta\ell}) \\
&\geq \psi(\hat{\mathbf{y}}^{\beta\ell}) + f_\beta(\ell) \quad \text{(by Claim 5.2)} \\
&= \psi(\hat{\mathbf{y}}^{\beta\ell}) + \mathrm{C}(\mathbf{y}^*) \\
&\geq \psi(\mathbf{y}^*) + \mathrm{C}(\mathbf{y}^*) \quad \text{(by Claim 5.1)} \\
&= F(\mathbf{y}^*).
\end{aligned}$$

■

Although we had initially designed the α-pass algorithm for MAX potentials, a similar argument can be used to show that α-pass performs exact clique inference for *any* arbitrary symmetric potential when we have two labels.

Claim 5.3 α*-pass is exact for arbitrary symmetric potentials over two labels.*

Proof Let the MAP $\mathbf{y}^*$ have label counts n_1 and $n-n_1$, and a vertex score of $\psi(\mathbf{y}^*)$. Let $\hat{\mathbf{y}}$ be the labeling returned by α-pass. Then we get $F(\hat{\mathbf{y}}) \geq F(\hat{\mathbf{y}}^{1n_1}) = \psi(\hat{\mathbf{y}}^{1n_1}) + C(n_1, n-n_1) \geq \psi(\mathbf{y}^*) + C(n_1, n-n_1) = F(\mathbf{y}^*)$. Since C is arbitrary, the result follows. ■

We also note that in some real-life tasks the vertex terms tend to heavily dominate the clique term. In such a case too, α-pass will provide an exact solution. This follows immediately from Claim 5.1 and the observation that $F(\mathbf{y}) \approx \psi(\mathbf{y})$.

5.2 Clique Inference for SUM Potentials

We focus on the Potts potential, the most popular member of the SUM family. Potts potential is given by $C^{\text{Potts}}(\mathbf{y}) = \lambda \sum_y n_y^2$ and the clique inference objective is:

$$\max_{y_1,\ldots,y_n} \sum_{j=1}^{n} \psi_{jy_j} + \lambda \sum_y n_y^2. \tag{7}$$

In the above, $\lambda > 0$ since we are interested in the associative case which encourages agreement among labels. It is well known that inference with Potts potentials over general graphs, is NP-hard (Boykov et al., 2001) when number of labels is > 2 even with $\lambda > 0$. So we will first show that the task remains NP-hard even for cliques. Then, we will prove that α-pass achieves an approximation ratio of $\frac{4}{5}$, followed by a more complex proof for a better and tight approximation ratio of $\frac{13}{15}$. We will then generalize the α-pass algorithm, which will result in an improved ratio of $\frac{8}{9}$ at a cost of higher runtime. In contrast, the α-expansion algorithm of Boykov et al. (2001) will be shown to have a ratio of only $\frac{1}{2}$.

Theorem 5 *When $C(\mathbf{y}) = \lambda \sum_y n_y^2, \lambda > 0$, clique inference is NP-hard.*

Proof We prove hardness by reduction from the NP-complete *Exact Cover by 3-sets* problem (Papadimitriou and Steiglitz, 1982). In an instance of Exact Cover by 3-sets, we are given a universe U of elements, a set S of subsets of U where each subset has three elements, and the goal is to find $S' \subseteq S$ that covers U, while minimizing $|S'|$. We create an instance of clique inference as follows. We let elements of U correspond to vertices, and each set in S to a label. Assign $\psi_{iy} = 2n\lambda$ if element i belongs to set y, and zero otherwise (set $\lambda > 0$ arbitrarily). Consider the problem of deciding if exactly $\frac{n}{3}$ out of m subsets cover U. The MAP score in the constructed clique inference instance will be $(2n^2 + 3^2\frac{n}{3})\lambda$ iff we can find an exact cover. ■

The above proof establishes that there cannot be an algorithm that is polynomial in both n and m. But we have not ruled out algorithms with complexity that is polynomial in n but exponential in m, say of the form $O(2^m n^k)$ for a constant k.

We next analyze the approximation guarantees provided by the α-pass algorithm. We first present an easy proof for a weaker bound of $\frac{4}{5}$ and then move on to a more detailed proof for $\frac{13}{15}$. Recall that the optimal labeling is $\mathbf{y}^*$ and the labeling output by α-pass is $\hat{\mathbf{y}}$.

Theorem 6 $F(\hat{\mathbf{y}}) \geq \frac{4}{5} F(\mathbf{y}^*)$.

Proof Without loss of generality assume that the label counts of $\mathbf{y}^*$ are $n_1 \geq n_2 \geq \ldots \geq n_m$. Then $\mathrm{C}(\mathbf{y}^*) = \lambda \sum_y n_y^2 \leq \lambda n_1 \sum_y n_y = \lambda n n_1$.

$$\begin{aligned} F(\hat{\mathbf{y}}) &\geq F(\hat{\mathbf{y}}^{1n_1}) = \psi(\hat{\mathbf{y}}^{1n_1}) + \mathrm{C}(\hat{\mathbf{y}}^{1n_1}) \\ &\geq \psi(\mathbf{y}^*) + \mathrm{C}(\hat{\mathbf{y}}^{1n_1}) \quad \text{(from Claim 5.1)} \\ &\geq \psi(\mathbf{y}^*) + \lambda n_1^2 \quad \text{(since } \lambda > 0\text{)} \\ &\geq \psi(\mathbf{y}^*) + \mathrm{C}(\mathbf{y}^*) - \lambda n_1 n + \lambda n_1^2 \\ &\geq F(\mathbf{y}^*) - \lambda n^2/4. \end{aligned} \tag{8}$$

Now consider the two cases where $F(\mathbf{y}^*) \geq \frac{5}{4}\lambda n^2$ and $F(\mathbf{y}^*) < \frac{5}{4}\lambda n^2$. For the first case we get from above that $F(\hat{\mathbf{y}}) \geq F(\mathbf{y}^*) - \lambda n^2/4 \geq \frac{4}{5}F(\mathbf{y}^*)$. For the second case, we know that the score $F(\hat{\mathbf{y}}^{mn})$ where we assign all vertices the last label is at least λn^2 (because all vertex scores are assumed to be non-negative) and thus $F(\hat{\mathbf{y}}) \geq \frac{4}{5}F(\mathbf{y}^*)$. ■

We stress that non-negativity of the vertex scores is important for the multiplicative bound of $\frac{4}{5}$, else one can construct examples where $F(\mathbf{y}^*) = 0$. At the same time, Equation 8 provides a useful additive bound that holds for arbitrary vertex scores.

We now state the more involved proof for showing that α-pass actually provides a tight approximation bound of $\frac{13}{15}$ for clique inference with Potts when the vertex scores are non-negative.

Theorem 7 *$F(\hat{\mathbf{y}}) \geq \frac{13}{15}F(\mathbf{y}^*)$. Further, this ratio is tight.*

Proof See Appendix A. ■

We next present a generalization of the α-pass algorithm that provides provably better guarantees for Potts.

5.2.1 GENERALIZED α-PASS ALGORITHM

In α-pass, for each label α, we go over each count k and find the best vertex score with k vertices assigned label α. We generalize this to go over all label subsets of size no more than q, a parameter of the algorithm that is fixed based on the desired approximation guarantee.

For each label subset $A \subseteq \{1,\ldots,m\}$ of size no more than q, and for each count k, maximize vertex scores with exactly k vertices assigned a label from A. For this, we sort the vertices in decreasing order of $\max_{\alpha \in A} \psi_{i\alpha} - \max_{y \notin A} \psi_{iy}$, assign the top k vertices their best label in A and the remaining their best label not in A. The best solution over all combinations (A,k) with $|A| \leq q$ is returned as the final labeling $\hat{\mathbf{y}}$. It is easy to see that this algorithm reduces to α-pass when $q = 1$.

The complexity of this algorithm is $O(nm^q \log n)$ because there are $\binom{m}{q}$ choices for the set A. In practice, we can use heuristics to prune the number of label subsets. Further, we can make the following claims about the quality of its output.

Theorem 8 *Generalized α-pass enjoys an approximation bound of $\frac{8}{9}$, that is, $F(\hat{\mathbf{y}}) \geq \frac{8}{9}F(\mathbf{y}^*)$.*

Proof The bound is achieved with $q = 2$. We provide the details in Appendix A. ■

We *conjecture* that the bound for general q is $\frac{4q}{4q+1}$. This bound is not tight as for $q = 1$ we have already shown that the $\frac{4}{5}$ bound can be tightened to $\frac{13}{15}$. With $q = 2$ we get a bound of $\frac{8}{9}$ which is better than $\frac{13}{15}$.

5.2.2 α-EXPANSION ALGORITHM

In general graphs, a popular method that provides the approximation guarantee of 1/2 for the Potts model is the graph-cut based α-expansion algorithm (Boykov et al., 2001). We explore the behavior of this algorithm for clique inference with Potts potentials.

In this scheme, we start with any initial labeling—for example, all vertices are assigned the first label, as suggested in Boykov et al. (2001). Next, for each label α we perform an α-expansion phase where we switch the labeling of an optimal set of vertices from their current label to α. We repeat this in a round over the m labels, until no vertices switch their labeling in a complete round.

For graphs whose edge potentials form a metric, an optimal α-expansion move is based on the use of the mincut algorithm of Boykov et al. (2001) which for the case of cliques can be $O(n^3)$. We next show how to perform optimal α-expansion moves more efficiently (in $O(mn^2)$ time) for all kinds of SUM potentials.

An α-expansion move: Let $\tilde{\mathbf{y}}$ be the labeling at the start of this move. For each label $y \neq \alpha$ create a sorted list S_y of vertices assigned y in $\tilde{\mathbf{y}}$ in decreasing order of $\psi_{i\alpha} - \psi_{iy}$. If in an optimal move, we move k_y vertices from y to α, then it is clear that we need to pick the top k_y vertices from S_y. Let r_i be the rank of a vertex i in S_y. Our remaining task is to decide the optimal number k_y to take from each S_y. We find these using dynamic programming. Without loss of generality assume $\alpha = m$. Let $D_j(k)$ denote the best score when k vertices with current labels in $1 \ldots j$ switch to α. We compute

$$D_j(k) = \max_{l \leq k,\ l \leq n_j(\tilde{\mathbf{y}})} D_{j-1}(k-l) + f_j(n_j(\tilde{\mathbf{y}}) - l) + \sum_{i':r_{i'} \leq l} \psi_{i'\alpha} + \sum_{i':r_{i'} > l} \psi_{i'j},$$

where $n_j()$ is the usual cardinality of label j in the labeling. Now we can find the optimal number of vertices to switch to α as $\text{argmax}_{k \leq n - n_\alpha(\tilde{\mathbf{y}})} D_{m-1}(k) + f_\alpha(k + n_\alpha(\tilde{\mathbf{y}}))$.

5.2.3 COMPARISON WITH EXISTING APPROXIMATION BOUNDS

As mentioned earlier, the C^{Potts} clique potential is equivalent to the sum of Potts potentials over edges of a complete graph. For arbitrary graphs with homogeneous Potts potential on edges, the alpha expansion algorithm of Boykov et al. (2001) and the LP relaxation algorithm of Kleinberg and Tardos (2002) provide a factor of 2 approximation guarantee for a minimization version of the objective. For cliques with homogeneous edge potentials, their objective reduces to an energy-based formulation (using their notation):

$$\min_{\mathbf{y}} \sum_j \theta_j(y_j) + \gamma \sum_{i,j>i} \delta(y_j \neq y_i) \equiv \min_{\mathbf{y}} \sum_j \theta_j(y_j) + \frac{\gamma}{2}(n^2 - \sum_y n_y^2), \tag{9}$$

where $\theta_j(y_j)$ denotes node energy and is assumed to be positive and γ denotes the homogeneous Potts parameter.

First we show that α-expansion provides an approximation bound of 2 even for the special case of cliques. Symmetrically, we also show that the α-expansion algorithm provides a bound of $\frac{1}{2}$ for our original max-version of the clique inference problem. Next, we show that α-pass provides a bound of $\frac{3}{2}$ for even the minimization objective above. Thus, for both the minimization and maximization objectives, α-pass improves upon the guarantees obtained by existing algorithms.

Theorem 9 *The α-expansion algorithm provides no better approximation guarantee than 2 for the minimization objective (9).*

Proof An instance where this occurs is as follows. There are n nodes and $n+1$ labels $y_0, \ldots, y_n$. The vertex energy for (x_i, y_0) is $2n-2$ and for (x_i, y_i) it is 0. All other vertex energies are ∞, and $\gamma = 1$. Suppose initially all nodes are assigned label y_0. Then the clique energy is 0 and vertex energy is $n(2n-2)$. Now considering any other label $y_i, i >= 1$, the optimal set of nodes to switch is just x_i. But this reduces vertex energy by $2n-2$ but increases clique energy by the same amount, so no switching is done. However the optimal solution has x_i labeled with y_i, which has a total energy of $n^2 - n$. ■

Theorem 10 *The α-expansion algorithm provides no better approximation guarantee than 1/2 for the maximization objective (7).*

Proof Consider an instance where $m = \sqrt{n}+1$, and $\lambda = 1$. Let $\psi_{u1} = 2\sqrt{n}$ for all u. Divide the vertices into $\sqrt{n}$ groups of size $\sqrt{n}$ each, and let $\psi_{u,i+1} = 2n$ for every vertex u in the i^{th} group. All other vertex scores are zero. Consider the solution where every vertex is assigned label 1. This labeling is locally optimal wrt any α-expansion move, and its score is $n^2(1+2/\sqrt{n})$. However, the exact solution assigns every vertex group its label, with a score $n^2(2+1/\sqrt{n})$, thus giving a ratio of $1/2$ in the limit. ■

Theorem 11 *The α-pass algorithm achieves an approximation ratio of $\frac{3}{2}$ for the minimization objective 9.*

Proof Suppose $k = \max_y n_y$ is the highest count in the optimal labeling. Consider two cases, $k \geq n/2$ and $k < n/2$.

If $k \geq n/2$, the clique energy is at least $\gamma(n^2 - (k^2 + (n-k)^2)) = \gamma 2k(n-k)$. The clique energy in α-pass is at most $\gamma(2n^2 - k^2 - (n-k)) = \gamma(n-k)(n+k-1)$. The vertex energy in the optimal labeling cannot be smaller than that in α-pass. Since $k >= n/2, (n+k-1)/2k \leq 3/2$.

If $k \leq n/2$, then the clique energy in the optimal labeling is at least $\gamma(n^2 - nk) = \gamma n(n-k)$. Therefore the ratio is again at most $(n+k-1)/n \leq 3/2$. ■

As expected, generalized α-pass provides a superior approximation ratio for Objective 9.

Theorem 12 *The generalized α-pass algorithm with two labels ($q = 2$) achieves an approximation ratio of $\frac{1+\sqrt{2}}{2}$ for objective 9.*

Proof We omit the complete proof as it is quite detailed and give only a brief sketch. Wlog assume that 1 and 2 are the two most dominant labels in the optimal labeling OPT, with counts $n_1, n_2 \leq n_1$. Consider two solutions—one given by α-pass (i.e., $q = 1$) when label 1 has count n_1, and another given by generalized α-pass with $q = 2$ when counts of labels 1 and 2 are $n_1 + n_2$. The node energies of these two labelings cannot be worse than that of OPT (Claim 5.1). So the approximation bound depends only on the ratios of the clique energies. Therefore wlog assume that $\gamma = 2$ in Objective 9. The clique energies of the two solutions cannot exceed $(n^2 - n_1^2)$ and $(n^2 - \frac{(n_1+n_2)^2}{2})$ respectively. We can now work out the two cases whether $n_1^2 \leq (n_1+n_2)^2/2$ or not, and get the desired approximation bound through contradictions. The analysis also shows that the bound is tight, and is achieved when

$n_1 = n/\sqrt{2}$ and $n_2 = n - n/\sqrt{2}$. ■

5.2.4 ENTROPY POTENTIALS AND THE α-PASS ALGORITHM

As an aside, let us explore the behavior of α-pass on another family of additive potentials—entropy potentials. Entropy potentials are of the form:

$$C(\mathbf{n}(\mathbf{y})) = \lambda \sum_y n_y \log n_y, \text{ where } \lambda > 0.$$

The main reason α-pass provides a good bound for Potts potentials is that it guarantees a clique potential of at least n_1^2 where n_1 is the count of the most dominant label in the optimal solution $\mathbf{y}^*$. The quadratic term compensates for possible sub-optimality of counts of other labels. If we had a sub-quadratic term instead, say $n_1 \log n_1$ for the entropy potentials, the same bound would not have held. In fact the following theorem shows that for entropy potentials, even though α-pass guarantees a clique potential of at least $n_1 \log n_1$, that is not enough to provide a good approximation ratio.

Theorem 13 *α-pass does not provide a bound better than $\frac{1}{2}$ for entropy potentials.*

Proof Consider a counter example where there are $m = n + \log n$ labels. Divide the labels into two sets—A with $\log n$ labels and B with n labels. The vertex scores are as follows: the vertices are divided into $\log n$ chunks of size $n/\log n$ each. If the j^{th} vertex lies in the y^{th} chunk, then let it have a vertex score of $\log n$ with label y in A and a vertex score of $\log n + \varepsilon$ with the j^{th} label in B. Let all other vertex scores be zero. Also, let $\lambda = 1$.

Consider the labeling which assigns the y^{th} label in A to the y^{th} chunk. Its score is $2n \log n - n \log \log n$. Now consider α-pass, with $\alpha \in A$. Initially vertex y will be set to the y^{th} label in B. The best labeling found by α-pass will assign every vertex to α, for a total score of roughly $n + n \log n$. If $\alpha \in B$, then again the best labeling will assign everything to α for a total score of roughly $(n+1)\log n$.

Thus the bound is no better than $\frac{1}{2}$ as $n \to \infty$. ■

Thus, α-pass provides good approximations when the clique potential is heavily dominated by the most dominating label. We now look at MAJORITY potentials, which are linear in the counts $\{n_y\}_y$. Looking at Theorem 13, we expect that α-pass will not have decent approximation guarantees for MAJORITY. This is indeed the case. We will prove in Section 5.3 that neither α-pass nor a natural modification of α-pass enjoy good approximation guarantees.

5.3 Clique Inference for MAJORITY Potentials

Recall that MAJORITY potentials are of the form $C = f_a(\mathbf{n}),\ a = \text{argmax}_y\ n_y$. We consider linear majority potentials where $f_a(\mathbf{n}) = \sum_y w_{ay} n_y$. The matrix $W = \{w_{yy'}\}$ is not necessarily diagonally dominant or symmetric.

We show that exact MAP for linear majority potentials can be found in polynomial time. We also present a modification to the α-pass algorithm to serve as an efficient heuristic, but without approximation guarantees. Then we present a Lagrangian relaxation based approximation, whose runtime is competitive with α-pass, but which provides much more accurate solutions.

5.3.1 MODIFIED α-PASS ALGORITHM

Assume that we magically know the majority label α in advance. Then for linear majority potentials, we can incorporate the linear clique term $w_{\alpha y} n_y$ in the various vertex scores, and this leads to the following natural modifications to the α-pass algorithm: (a) While making iterations for the label α, sort the vertices according to the modified metric $\psi_{i\alpha} + w_{\alpha\alpha} - \max_{y \neq \alpha}(\psi_{iy} + w_{\alpha y})$, and (b) While sweeping the list for α with varying values of k, discard all candidate solutions whose majority label is not α.

However even after these modifications, α-pass does not provide the same approximation guarantee as for homogeneous Potts potentials, as we prove next.

Theorem 14 *The modified α-pass algorithm cannot have an approximation ratio better than $\frac{1}{2}$ on linear majority potentials with arbitrary W.*

Proof Consider the degenerate example where all vertex scores are zero. Let β and γ be two fixed labels and let the matrix W be defined as follows: $w_{\beta\gamma} = M + \varepsilon$, $w_{\beta y} = M \; \forall y \neq \beta, \gamma$ and all the other entries in W are zero.

In modified α-pass, when $\alpha \neq \beta$, the labeling returned will have a zero score. When $\alpha = \beta$, all vertices will prefer the label γ, so α-pass will have to assign exactly $n/2$ vertices to β to make it the majority label, thus returning a score of $\frac{(M+\varepsilon)n}{2}$. However, consider the labeling which assigns n/m vertices to each value, with a score of $(m-1)Mn/m$. Hence the approximation ratio cannot be better than $\frac{1}{2}$. ■

5.3.2 EXACT ALGORITHM

Since MAJORITY potentials are linear, we can pose the optimization problem in terms of Integer Programs (IPs). Assume that we know the majority label α. Then, the optimization problem corresponds to the IP:

$$\begin{aligned} \max_{\mathbf{z}} \sum_{i,y} (\psi_{iy} + w_{\alpha y}) z_{iy}, \\ \text{s.t. } \forall y: \; \sum_i z_{iy} \leq \sum_i z_{i\alpha}, \\ \forall i: \; \sum_y z_{iy} = 1, \; z_{iy} \in \{0,1\}. \end{aligned} \tag{10}$$

We can solve m such IPs by guessing various labels as the majority label, and reporting the best overall labeling as the output. However, Equation 10 cannot be tightly relaxed to a linear program. This can be easily shown using a counter example: Consider a 3-vertex, 3-label clique with a zero W matrix. Let the vertex score vectors be $\psi_0 = (1,4,0)$, $\psi_1 = (4,0,4)$, $\psi_2 = (3,4,0)$. While solving for $\alpha = 0$, the best IP labeling is $1,0,0$ with a score of 11. However the LP relaxation has the solution $\mathbf{z} = (0,1,0;1,0,0;1/2,1/2,0)$ with a score of 11.5.

This issue can be resolved by making the constraint matrix totally unimodular as follows. Guess the majority label α, its count $k = n_\alpha$, and solve the following IP:

$$\begin{aligned} \max_{\mathbf{z}} \sum_{i,y} (\psi_{iy} + w_{\alpha y}) z_{iy}, \\ \text{s.t. } \forall y \neq \alpha : \sum_i z_{iy} \leq k, \\ \sum_i z_{i\alpha} = k, \\ \forall i : \sum_y z_{iy} = 1,\ z_{iy} \in \{0,1\}. \end{aligned} \tag{11}$$

This IP solves the degree constrained bipartite matching problem, which can be solved exactly in polynomial time. Indeed, it can be shown that the constraint matrix of this IP is totally unimodular, so its LP relaxation will have an integral solution. We refer the reader to Gupta et al. (2009) for the details. Thus we solve $O(mn)$ such problems by varying α and k, and report the best solution. We believe that the above LP is concave in k, so the system for a fixed α should be efficiently solvable using golden section search.

5.3.3 LAGRANGIAN RELAXATION BASED ALGORITHM FOR MAJORITY POTENTIALS

Solving the linear system in Equation 11 is very expensive because we need to solve $O(mn)$ LPs, whereas the system in Equation 10 cannot be solved exactly using a linear relaxation. Here, we look at a Lagrangian Relaxation based approach (LR), where we solve the system in Equation 10 but bypass the troublesome constraint $\forall y \neq \alpha : \sum_i z_{iy} \leq \sum_i z_{i\alpha}$.

We use Lagrangian relaxation to move this constraint to the objective function. Any violation of this constraint is penalized by a positive penalty term. Consider the following modified program, also called the Lagrangian:

$$\begin{aligned} L(\gamma) = L(\gamma_1, \ldots, \gamma_m) = \max_{\mathbf{z}} \sum_{i,y} (\psi_{iy} + w_{\alpha y}) z_{iy} \quad + \quad \sum_y \gamma_y (\sum_i z_{i\alpha} - \sum_i z_{iy}), \\ \text{s.t. } \forall i : \sum_y z_{iy} = 1, \qquad z_{iy} \in \{0,1\}. \end{aligned} \tag{12}$$

For $\gamma \geq \mathbf{0}$, and feasible $\mathbf{z}$, $L(\gamma)$ is an upper bound for our objective in Equation 10. Thus, we compute the lowest such upper bound:

$$L^* = \min_{\gamma \geq \mathbf{0}} L(\gamma). \tag{13}$$

Further, the penalty term in Equation 12 is linear in $\mathbf{z}$, so we can merge it with the first term to get another set of modified vertex potentials:

$$\psi^\alpha_{iy} \triangleq \psi_{iy} + w_{\alpha y} - \gamma_y + \begin{cases} \sum_{y'} \gamma_{y'} & y = \alpha \\ 0 & y \neq \alpha \end{cases}.$$

Equation 12 can now be rewritten in terms of ψ^α, with the only constraint that $\mathbf{z}$ correspond to a valid labeling:

$$\begin{aligned} \max_{\mathbf{y}} \sum_{i,y} \psi^\alpha_{iy} z_{iy}, \\ \text{s.t. } \forall i : \sum_y z_{iy} = 1,\ \ z_{iy} \in \{0,1\}. \end{aligned}$$

Hence, $L(\gamma)$ can be computed by independently assigning each vertex i to its best label viz. $\text{argmax}_y \psi^{\alpha}_{iy}$.

We now focus on computing L^*. We use an iterative approach, beginning with $\gamma = \mathbf{0}$, and carefully choose a new γ at each step to get a non-increasing sequence of $L(\gamma)$'s. We describe the method of choosing a new γ later in this section, and instead outline sufficient conditions for termination and detection of optimality.

Theorem 15 *$\mathbf{z}^*$ and γ^* are optimum solutions to Equations 10 and 13 respectively if they satisfy the conditions:*

$$\forall y: \quad \sum_i z^*_{iy} \leq \sum_i z^*_{i\alpha}, \tag{14}$$

$$\forall y: \quad |\gamma^*_y(\sum_i z^*_{iy} - \sum_i z^*_{i\alpha})| = 0. \tag{15}$$

Theorem 15 holds only for fractional $\mathbf{z}^*$. To see how, consider an example with three vertices and two labels. Let $\psi_{i1} + w_{\alpha 1} > \psi_{i2} + w_{\alpha 2}$ for all i and α. During Lagrangian relaxation with $\alpha = 2$, initially $\gamma = \mathbf{1}$ will cause all vertices to be assigned label 1, violating Equation 14. Since the count difference $\sum_i z_{i1} - \sum_i z_{i2} \in \{\pm 1, \pm 2, \pm 3\}$, any non-zero γ_1 will violate Equation 15. Subsequent reduction of γ_1 to zero will again cause the original violation of Equation 14. Consequently, one of Equations 14 and 15 will never be satisfied and the algorithm will oscillate.

To tackle this, we relax Equation 15 to $|\gamma_y(\sum_i z^*_{iy} - \sum_i z^*_{i\alpha})| \leq \varepsilon$, where ε is a small fraction of an upper bound on γ_y. This helps in reporting labelings that respect the majority constraint in Equation 14 and are close to the optimal.

The outline of the algorithm is described in Figure 2.

We now discuss a somewhat conservative approach to select a new γ at every step. We initially attempted subgradient optimization and golden search to compute step direction and sizes for changing γ. However, we ran into various practical difficulties. Subgradient optimization required very careful tweaking of the step size across iterations, an issue exacerbated by the discrete nature of our problem. On the other hand, golden search was too aggressive in practice, leading to many avoidable label flips and consequently many more iterations. So instead we implemented a conservative approach which we describe next.

5.3.4 CONSERVATIVE COORDINATE DESCENT

We perform conservative coordinate descent which avoids large changes and thus too many label flips. Let y be the worst violating label in the current iteration. We will first consider the case when its count exceeds that of α, so that Equation 14 does not hold.

To decrease the count of y, we need to increase γ_y. Let i be a vertex currently assigned y and let $\beta(i)$ be its second most preferred label under the vertex potentials ψ^{α}_i. The vertex $j = \text{argmax}_{i:z_{iy}=1} \psi^{\alpha}_{i\beta(i)} - \psi^{\alpha}_{iy}$ is the easiest to flip. So we increase γ_y just enough to make this flip happen. The new value of γ_y is therefore given by:

$$\gamma_y = \min_{i:z_{iy}=1} \begin{cases} \Delta\psi(i,y,\beta(i)) + \gamma_{\beta(i)} & \beta(i) \neq \alpha \\ \frac{1}{2}(\Delta\psi(i,y,\alpha) - \sum_{y' \neq y} \gamma_{y'}) & \beta(i) = \alpha \end{cases}, \tag{16}$$

where $\Delta\psi(i,y,y')$ denotes $\psi_{iy} + w_{\alpha y} - \psi_{iy'} - w_{\alpha y'}$. It is possible that by flipping vertex j, $\beta(j)$ now violates Equation 14. Moreover, increasing γ_y also increases $\psi^{\alpha}_{i\alpha}$, so some other vertices that are not

assigned y may also move to α. However since the change is conservative, we expect this behavior to be limited. In our experiments, we found that this conservative scheme converges much faster than golden search over a variety of data.

We now look at the case when Equation 14 is satisfied by all labels but Equation 15 is violated by some label y. In this scenario, we need to decrease γ_y to decrease the magnitude of the violation. Here too, we conservatively decrease γ_y barely enough to flip one vertex to y. If i is any vertex not assigned label y and $\beta(i)$ is its current label, then the new value of γ_y is given by:

$$\gamma_y = \max_{i:z_{iy}\neq 1} \begin{cases} \Delta\psi(i,y,\beta(i)) + \gamma_{\beta(i)} & \beta(i) \neq \alpha \\ \frac{1}{2}(\Delta\psi(i,y,\alpha) - \sum_{y'\neq y}\gamma_{y'}) & \beta(i) = \alpha \end{cases} . \tag{17}$$

Note that the arguments of Equations 16 and 17 are the same. In this case too, in spite of a conservative move, more than one vertex marked α may flip to some other value, although at most one of them will be flipped to y. As before, the small magnitude of the change restricts this behavior in practice.

Input: ψ, W, α,maxIters,tolerance
Output: approximately best assignment $\hat{\mathbf{y}}$
$\gamma \leftarrow \mathbf{0}$;
iter $\leftarrow 0$;
$\hat{\mathbf{z}} \leftarrow$ Assignment with all vertices assigned α;
while *iter* $<$ *maxIters* **do**
  Compute $L(\gamma)$ (Equation 12), let $\mathbf{z}$ be the solution;
  if $F(\mathbf{z}) > F(\hat{\mathbf{z}})$ **then**
    $\hat{\mathbf{z}} \leftarrow \mathbf{z}$;
  end
  $(y, \Delta) \leftarrow$ Worst violator and violation (Equations 14 and 15);
  if $\Delta <$ *tolerance* **then**
    We are done, $L^* = L(\gamma)$;
    break;
  else
    Modify γ_y using conservative coordinate descent;
  end
  iter $\leftarrow$ iter+1;
end
Construct assignment $\hat{\mathbf{y}}$ from $\hat{\mathbf{z}}$;
return $\hat{\mathbf{y}}$

Algorithm 2: LR Algorithm for Majority potentials

5.4 Computing All Node Max-marginals

In this section, we discuss an important optimization that speeds up message-passing in a cluster graph. We show that for symmetric potentials we can compute max-marginals in $O(m^2)$ calls to the clique inference algorithm in practice, as opposed to the expected nm invocations. Cliques can be arbitrarily big, so removing the dependence on n is very helpful in practice.

Our basic strategy is as follows. For label pairs α, β, define a modified clique potential function $C_{\alpha\beta}$ that adds "1" to the count for label β and subtracts "1" from the count of label α:

$$C_{\alpha\beta}(\mathbf{y}) = C(n_1(\mathbf{y}), \ldots, n_\alpha(\mathbf{y}) - 1, \ldots, n_\beta(\mathbf{y}) + 1, \ldots n_m).$$

Find MAP labeling $\mathbf{y}^{\alpha\beta}$ using the modified scoring function $F_{\alpha\beta}(\mathbf{y}) = \psi(\mathbf{y}) + C_{\alpha\beta}(\mathbf{y})$. Let v be a vertex whose label in $\mathbf{y}^{\alpha\beta}$ is α, then the max marginal $M_{v\beta} \triangleq \max_{\mathbf{y}:y_v=\beta} F(\mathbf{y}) = F_{\alpha\beta}(\mathbf{y}^{\alpha\beta}) - \psi_{v\alpha} + \psi_{v\beta}$. The outgoing message $m_{t\to v}(\beta)$ is then simply $M_{v\beta} - m_{v\to t}(\beta)$ as per Equation 4. A proof of correctness of this optimization is as follows.

Theorem 16 $\max_{\mathbf{y}:y_v=\beta} F(\mathbf{y}) = F_{\alpha\beta}(\mathbf{y}^{\alpha\beta}) - \psi_{v\alpha} + \psi_{v\beta}$.

Proof

$$\begin{aligned}
\max_{\mathbf{y}} F_{\alpha\beta}(\mathbf{y}) &= \max_{\mathbf{y}:y_v=\alpha} F_{\alpha\beta}(\mathbf{y}) \text{ since } v \text{ is labeled } \alpha \text{ in } \mathbf{y}^{\alpha\beta} \\
&= \max_{\mathbf{y}:y_v=\alpha} \psi(\mathbf{y}) + C_{\alpha\beta}(\mathbf{y}) \\
&= \max_{\mathbf{y}:y_v=\beta} \psi(\mathbf{y}) - \psi_{v\beta} + \psi_{v\alpha} + C(\mathbf{y}) \\
&= \max_{\mathbf{y}:y_v=\beta} F(\mathbf{y}) - \psi_{v\beta} + \psi_{v\alpha}.
\end{aligned}$$

■

We invoke the above strategy for all label pairs α, β when some vertex in the original MAP gets label α. There is no guarantee that all *nm* messages would be generated by the above scheme. The ones that are not covered are computed using separate invocations of MAP.

This concludes the description of our various clique inference algorithms and their theoretical properties. The discussion until now had focused on the traditional unigram-clique collective inference model. We now switch tracks and describe an extension of the collective inference framework—one which captures richer kinds of associativity that is often present in the data. We will see that the same cluster message passing setup and collective inference algorithms can be used almost as is in this more general scenario.

6. Properties-based Collective Inference Framework

We broaden the notion of collective inference to encourage richer forms of associativity amongst the labelings of multiple records. This more general framework has applications in domain adaptation. We illustrate this via an example.

Example: Consider extracting bibliographic information from an author's publications homepage using a model trained on a different set of pages. The labels of interest are Title, Author, Venue, and Date. Typically, within each homepage (a domain) we expect consistency in the style of individual publication records. For example, we expect the following properties to be largely uni-valued *inside* a domain:

1. The ordering of labels in the labelings (e.g., Title $\to$ Author* $\to$ Venue).
2. The token preceding the Title, or 'Start' if Title occurs at the beginning.
3. The label appearing after the Venue, or 'End' if Venue occurs at the end.
4. Label of the token 'IEEE'.

A Simulator for estimating Railway Line Capacity. *In APORS - 2003*.	(Start, Date, <b>)	Bhardwaj, P. (2001). Delegating Pricing Decisions. *Marketing Science* 20(2). 143-169.	('.', Volume,)
Scheduling Loosely Connected Task Graphs. *Journal of Computer and System Sciences , August 2003 .*	(Start, Date, <b>)	Balasubramaniam, S. and P. Bhardwaj (2004). When not all conflict is bad: Manufacturing marketing conflict and strategic incentive design. *Management Science* 50(4). 489-502.	('.', Volume,)
Devanagari Pen-written Character Recognition. *In ADCOM - 2001 .*	(Start, Date, <b>)	Bhardwaj, P. and S. Balasubramaniam (2005). Managing Channel Profits: The Role of Managerial Incentives. Forthcoming *Quantitative Marketing and Economics*.	('.', End,)

Table 2: Two publications pages with different styles. Text in parentheses shows the values of three properties on the correct labelings: (i) Token before Title (ii) First non-Other label after Venue (iii) HTML tag containing Title. The properties are largely uni-modal inside a page, but the mode varies across pages.

Note that 'IEEE' will tend to recur across multiple records, so the last property corresponds to the unigram-based cliques that we modeled thus far. It is clear now that we can gain a lot more if we couple the records together according to 'richer' properties than just unigrams.

Of course the choice of properties is crucial. For all the properties illustrated above, we expect that the labelings inside the domain agree on the property value, without caring for what the value actually is (which varies from domain to domain). This allows us to use the same property on different domains, with varying formatting and authoring styles. Table 2 illustrates this for two publications pages with different styles. It shows three properties that take on largely similar values across records inside a domain, but the dominant value changes across domains. Thus we can reward associativity of these property values using the same symmetric potentials that we have used for unigram cliques.

Now assume that we have an array of such conformance-promoting properties, and a basic MRF trained on some labeled domains. An effective way of deploying this MRF on a previously unseen domain is by labeling the records in the new domain collectively while encouraging the individual labelings to agree on our set of properties. If the properties continue to remain associative in the unseen domain, then collective inference can be expected to correct a significant number of errors. This provides us with an inference-only approach, in contrast to many existing solutions for domain adaptation which require model re-training (Blei et al., 2002; Blitzer et al., 2006; Mann and McCallum, 2007).

We now give an intuitive description of how the introduction of properties causes only minor changes in our collective inference framework. The modified collective inference objective is now given by:

$$\arg\max_{\mathbf{y}^1,\ldots,\mathbf{y}^N} \left(\sum_{k=1}^{N} \sum_{i=1}^{|\mathbf{x}^k|} \phi_i^k(y_i^k, y_{i-1}^k) \right) + \sum_{\mathrm{g} \in \mathcal{G}} \mathrm{C_g}(\{\mathrm{g}(\mathbf{x}^k, \mathbf{y}^k)\}_k),$$

instead of Equation 2. Here $\mathcal{G}$ is the set of properties, each of which defines an associative clique. We reuse the cluster message passing setup in this collective inference scenario. We define one cluster per chain, one per property, while separators remain singleton—they correspond to the property value. The new messages are of the kind $m_{\mathrm{g}\to k}(v)$ and $m_{k\to \mathrm{g}}(v)$, where v is a possible value output by the property g(). The property-to-chain messages are computed as before using our potential-specific clique inference algorithms. However the chain-to-property messages $m_{k\to \mathrm{g}}$ cannot be com-

puted for arbitrary properties. To illustrate, say $g(\mathbf{x}^k, \mathbf{y}^k) =$ Number of tokens marked Author in $\mathbf{y}^k$. Computing chain messages for this property is quite expensive, if not hard. So for computational reasons we restrict ourselves to *Markovian properties*, that is, properties whose chain-to-property messages can be computed with a first-order algorithm like Viterbi or max-product. Examples of Markovian properties include—Token before Title, and Label after Author. A sample message for the second property would be, say, $m_{k\to g}(\text{Title})$, which can be computed by running Viterbi with the first-order constraint that Author be followed by Title. In general, values of Markovian properties depend only on the Markovian blanket of a part of the chain, which leads to tractability of message computation. We refer the reader to Gupta et al. (2009) for the formal technical details of chain-to-property message computation for Markovian properties.

We stress that such a support for Markovian properties is not possible in alternative collective inference approaches like TRW-S. This is because even with Potts potentials over properties, the inter-cluster separators still remain complete chain labelings instead of scalar property values, so the cluster model cannot be represented as a pairwise graphical model.

In Section 7.3 we will look at the effect of introducing associative properties in the context of domain adaptation on a citation extraction task.

7. Experiments

Our goal is to empirically demonstrate that cluster message passing is indeed a more accurate and efficient framework for doing collective inference. Once this is established, it would necessitate the design of fast and accurate message computation algorithms that work for symmetric clique potentials. This would justify our design and analysis of the various clique inference algorithms presented in this paper. After illustrating the effectiveness of cluster message passing, we will show that the extension of collective inference to capture richer associative properties leads to significant boosts in domain adaptation tasks. Keeping these goals in mind, we present results of three different experiments—clique inference, collective inference, and the extension to general Markovian properties.

First, in Section 7.1 we compare our clique inference algorithms against applicable alternatives in the literature. We compare the algorithms on computation speed and accuracy of the MAP assignments. For Potts potentials, we show that α-pass provides similar MAPs as the various alternatives but is up to two orders faster. For linear MAJORITY potentials (Equation 6), we compare our algorithms against the exact LP-based approach of Section 5.3.2 and the ICM algorithm. For other clique potentials that are not decomposable along the clique edges, we compare α-pass against the ICM algorithm.

Second, in Section 7.2 we show that message passing on the cluster graph is a more effective method for collective inference than its alternatives. For collective models with Potts potentials, we compare cluster message passing against belief propagation that decomposes Potts along the clique edges. In the case of linear MAJORITY potentials, we compare against the stacking based approach of Krishnan and Manning (2006).

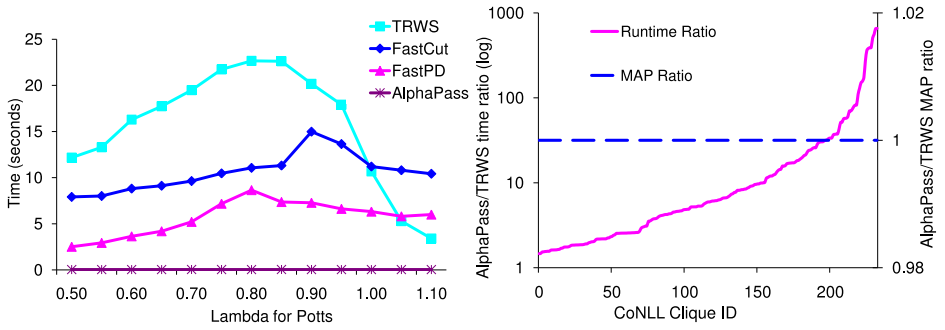
Finally, in Section 7.3 we demonstrate the application of our generalized collective framework on domain adaptation. On a citation extraction task, we show that capturing the associativity of richer properties leads to significant reduction in test error.

λ	ICM	α-pass	α^2-pass	α-exp DP	α-exp	FastCut	FastPD	TRW-S	DD
Clique MAP scores									
0.50	4866.7	4862.3	4864.2	4866.0	4867.0	4867.2	4866.7	4866.4	4833.3
0.55	4878.0	4872.6	4875.5	4878.5	4879.6	4879.7	4879.1	4878.6	4838.1
0.60	4899.8	4893.8	4895.8	4898.0	4900.9	4900.8	4900.7	4899.7	4865.6
0.65	4914.9	4909.4	4911.6	4913.3	4915.9	4916.9	4914.6	4915.1	4881.3
0.70	4919.6	4913.0	4916.7	4916.6	4921.5	4923.1	4920.6	4919.6	4891.2
0.75	4930.0	4934.7	4936.9	4928.8	4935.1	4936.8	4938.2	4938.4	4926.8
0.80	4965.0	4969.3	4972.5	4959.5	4959.5	4971.2	4954.3	4971.4	4961.7
0.85	4977.3	5009.6	5010.0	4995.5	4988.3	5002.8	4997.7	5007.0	4999.9
0.90	5018.5	5082.3	5082.3	5073.9	5081.2	5081.1	5080.1	5079.0	5049.1
0.95	5053.5	5155.9	5155.9	5154.4	5154.9	5155.1	5154.9	5150.7	5100.9
1.00	5137.2	5264.3	5264.3	5264.3	5264.3	5264.3	5264.3	5262.1	5161.9
1.05	5279.4	5417.1	5417.1	5417.1	5417.1	5417.1	5417.1	5417.1	5269.5
1.10	5383.8	5528.1	5528.1	5528.1	5528.1	5528.1	5528.1	5528.1	5358.1
All	65223.5	65812.5	65831.2	65794.0	65813.5	65844.2	65816.6	65833.4	65137.3
Running Time (ms)									
0.50	24	38	464	155	5900	7910	2510	12170	117180
0.55	27	38	455	174	6610	8010	2930	13290	117480
0.60	28	38	454	160	8140	8810	3640	16290	118270
0.65	29	38	452	157	8700	9120	4190	17740	117050
0.70	35	43	470	162	11680	9630	5210	19480	118360
0.75	34	41	460	155	13770	10460	7180	21730	117170
0.80	42	39	459	153	16710	11060	8650	22660	117450
0.85	38	44	464	139	18260	11310	7360	22610	117730
0.90	42	42	462	163	19100	14980	7280	20170	117670
0.95	49	39	458	127	16450	13640	6620	17910	117070
1.00	58	39	456	85	11280	11200	6320	10670	117240
1.05	59	39	458	82	10730	10800	5810	5280	116780
1.10	50	39	467	82	10490	10420	6000	3400	116920
All	514	519	5979	1795	157820	137350	73700	203400	1526370

Table 3: Clique MAP scores and runtimes of various clique inference algorithms for Potts potential. Each number is an aggregate over 25 cliques for the corresponding λ.

7.1 Clique Inference Experiments

For clique potentials decomposable over clique edges, we compare our clique inference algorithms against sequential tree re-weighted message passing (TRW-S), graph-cut based inference (α-exp), the ICM algorithm, and recent advancements including the faster graph-cut algorithm of Alahari et al. (2008) (FastCut), the fast primal-dual algorithm of Komodakis and Tziritas (2007) (FastPD), and the dual-decomposition scheme of Komodakis et al. (2007a) (DD). For non-decomposable potentials, we present comparisons against the ICM algorithm. We present comparison results on running time and quality of the MAP. Our experiments were performed on both synthetic and real data.



(a) POTTS: Runtime vs λ (b) CoNLL with Potts: α-pass vs TRW-S

Figure 2: Potts potential: (a) Runtime of α-pass vs TRW-S, Graph-cut and FastPD aggregated on 25 cliques as λ is varied in the synthetic data set POTTS (b) Comparing α-pass and TRW-S on MAP scores and runtimes for each CoNLL clique.

Synthetic Data Set: We generated cliques with 100 vertices and $m = 24$ labels by choosing vertex potentials at random from $[0,2]$ for all vertex-label pairs. A Potts version (POTTS) was created by gradually varying λ, and generating 25 cliques for every value of λ. We also created analogous ENTROPY, MAKESPAN and MAKESPAN2 versions of the data set by choosing entropy ($\lambda \sum_y n_y \log n_y$), linear makespan ($\lambda \max_y n_y$) and square makespan ($\lambda \max_y n_y^2$) clique potentials respectively.

For linear MAJORITY potentials we generated two kinds of data sets (parameterized by λ): (a) MAJ-DENSE obtained by generating a random symmetric W for each clique, where $W_{yy} = \lambda$ was the same for all y and $W_{yy'} \in [0, 2\lambda]$ $(y \neq y')$, and (b) MAJ-SPARSE from symmetric W with $W_{yy'} \in [0, 2\lambda]$ for all y, y', roughly 70% of whose entries were zeroed. This sparse data set is supposed to mirror the sparseness of W in real-life data sets.

CoNLL Data Set: The CoNLL 2003 data set[1] is a popular choice for demonstrating the benefit of collective labeling in named entity recognition tasks. We used the BIOU encoding of the entities, that resulted in 20 labels. We took a subset of 1460 records from the test set of CoNLL, and selected all 233 cliques of size 10 and above. The smaller cliques were ignored as the algorithms hardly differ in performance over them. The median and largest clique sizes were 16 and 259 respectively. The vertex potentials of the cliques were set by a sequential Conditional Random Field trained on a separate training set. We created a Potts version by setting $\lambda = 0.9/n$ using the development set, where n is the clique size. Such a λ allowed us to balance the vertex and clique potentials for each clique. A majority version was also created by learning W discriminatively in the training phase.

We developed Java implementations for all our algorithms—α-pass, generalized α-pass with $q = 2$, dynamic programming based α-expansion of Section 5.2.2 (denoted α-exp DP), modified α-pass, Lagrangian relaxation, and the Exact LP-based algorithm for Majority potentials. We used publicly available C++ implementations for TRW-S,[2] (Boykov et al., 2001; Szeliski et al., 2006; Kolmogorov and Zabih, 2004; Boykov and Kolmogorov, 2004), FastCut (Alahari et al., 2008), and FastPD (Komodakis and Tziritas, 2007; Komodakis et al., 2008). In addition, we implemented DD

1. This data set can be found at `http://cnts.uia.ac.be/conll2003/ner/`.
2. This code can be found at `http://www.adastral.ucl.ac.uk/~vladkolm/papers/TRW-S.html` and `http://vision.middlebury.edu/MRF/`.

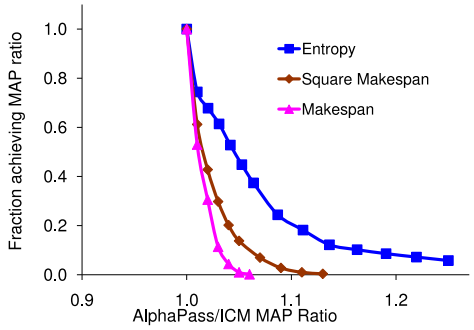


Figure 3: Comparing the MAP quality of α-pass vs ICM on non-decomposable potentials. Plotted point (r, f) denotes that for a fraction f of the cliques, α-pass returned a MAP score at least r times that of ICM.

in C++ and ICM in Java. All experiments were performed on a Pentium-IV 3.0 GHz machine with 8 GB of RAM.

7.1.1 EDGE DECOMPOSABLE POTENTIALS

Table 3 compares the various clique inference algorithms on the POTTS data set. We vary the Potts parameter λ uniformly in the range $[0.5, 1.1]$. This range is of special interest because it allows maximal contention between the clique and vertex potentials. For λ outside this range, the MAP is usually a trivial assignment, viz. one which either individually assigns each vertex to its best label (optimizing vertex potential), or assigns all vertices to a single label (optimizing clique potential). Each number in the upper half of Table 3 is the aggregate MAP score of 25 cliques for that particular λ, while the bottom half reports the aggregate clique inference time rounded to the nearest millisecond.

We observe that apart from ICM and DD, all the other algorithms return MAP scores in a small interval (0.5%) of each other. In particular α-pass performs almost identical to α-exp and FastPD in this regard. FastCut provides the best aggregate MAP scores, but α^2-pass and TRW-S are quite close as well. We note that the three expansion algorithms α-exp, α-exp DP, and FastCut return different MAP scores, primarily because they employ different bootstrapping steps. Further we observe that DD returns significantly low MAP scores, because even with suitable step-size selection heuristics such as the one proposed by Komodakis et al. (2007a), DD converges very slowly and invariably hits the maximum iteration limit of 30 with a sub-par MAP. ICM also returns lower MAP scores, as expected, due to its highly local updates.

In terms of runtime, Table 3 shows a clear separation of the algorithms. ICM and α-pass are the fastest, while α-exp, FastCut, FastPD, and TRW-S are 150-400 times slower. This is because in contrast to α-pass, the other algorithms perform multiple costly iterations (e.g., an iteration of TRW-S is $O(n^2)$). The cut-based algorithms and FastPD are faster than TRW-S, while DD is the slowest, at 3000 times α-pass. This is because DD has to deal with $O(n)$ spanning trees which is costly, and leads to slow convergence. Consequently DD hits the iteration limit every time. Our α^2-pass and dynamic programming based α-expansion (α-exp DP) algorithms are ten and three times slower than α-pass respectively, but still more than an order faster than the other algorithms. Finally we

see that while α-pass and α^2-pass take almost a constant time irrespectively of λ, the other schemes take up more time (as they iterate more) around the "maximal contention range" of $[0.7, 1.0]$ for λ. Figure 2(a) illustrates this behavior, where we vary λ and plot the runtimes of α-pass, FastPD, TRW-S, and FastCut. Note that α-pass lies very close to the x-axis, illustrating its small runtime.

Figure 2(b) presents the results on Potts cliques from the CoNLL data set. For simplicity we only compare α-pass with TRW-S as the other algorithms behave analogous to the synthetic scenario. For each clique, we plot (a) the ratio of the α-pass MAP score with that of TRW-S, and (b) ratio of TRW-S runtime vs α-pass runtime. While both the algorithms report the same MAP, α-pass is still more than 10 times faster on more than one-third of the cliques, and is never slower. This ratio is not as bad as for synthetic cliques mainly because the median clique size here is much smaller at 16.

7.1.2 NON-DECOMPOSABLE POTENTIALS

In this case, we cannot compare against the TRW-S or graph-cut based algorithms. Hence we compare with the ICM algorithm that has been popular in such scenarios (Lu and Getoor, 2003). The scheme of Potetz and Lee (2008) is another alternative, but we omit a comparison with it as it is bound to be quite expensive in m (see discussion in Section 8).

We varied λ with increments of 0.02 in $[0.7, 1.1)$ and generated 500 cliques each from MAJ-DENSE, MAJ-SPARSE, ENTROPY, MAKESPAN and MAKESPAN2. We measure the ratio of MAP score of α-pass with ICM and for each ratio r we plot the fraction of cliques where α-pass returns a MAP score at least r times that of ICM. Figure 3 shows the results on all the potentials except MAJORITY, which will be presented later. The curves for linear and square makespan lie totally to the right of $ratio = 1$, which is expected because α-pass will always return the true MAP for those potentials. In contrast ICM can only return a locally optimal solution. For entropy, α-pass was found to be significantly better than ICM in all the cases. The runtimes of ICM and α-pass were similar.

7.1.3 MAJORITY POTENTIALS

In Figures 4(a) and 4(b), we compare ICM, Lagrangian Relaxation (LR) and modified-α-pass (Section 5.3.1, denoted ModAlpha) against the LP-based exact method (LP) on synthetic data. Each curve plots, for each MAP ratio r, the fraction of cliques on which ICM (or LR or ModAlpha) returns a MAP score better than r times the optimal MAP score. An ideal algorithm's curve would just be a dot at $(1, 1)$ indicating that it retrieves the true MAP for all the cliques.

We observe that on MAJ-DENSE, both ModAlpha and ICM return a MAP score better than 0.85 of the true MAP, with ICM being slightly better. However, LR out-performs both of them, providing a MAP ratio always better than 0.97 and returning the true MAP in more than 70% of the cases. In MAJ-SPARSE too, LR dominates the other two algorithms, returning the true MAP in more than 80% of the cases, with a MAP ratio always better than 0.92. Further it can be derived that on average, LR returns a MAP score 1.15 times that of ICM. Thus, LR performs much better than its competitors across dense as well as sparse majority potentials.

The results on CoNLL data set, whose W matrix is 85% sparse, are displayed in Figure 4(c). ICM, ModAlpha, and LR return the true MAP in 87%, 95% and 99% of the cliques respectively, with the worst case MAP ratio of LR being 0.97 as opposed to 0.94 and 0.74 for ModAlpha and ICM respectively. Figure 4(d) displays runtime ratios on all CoNLL cliques for all three inexact algorithms vs LP. ICM and ModAlpha are roughly 100-10000 times faster than LP, while LR is

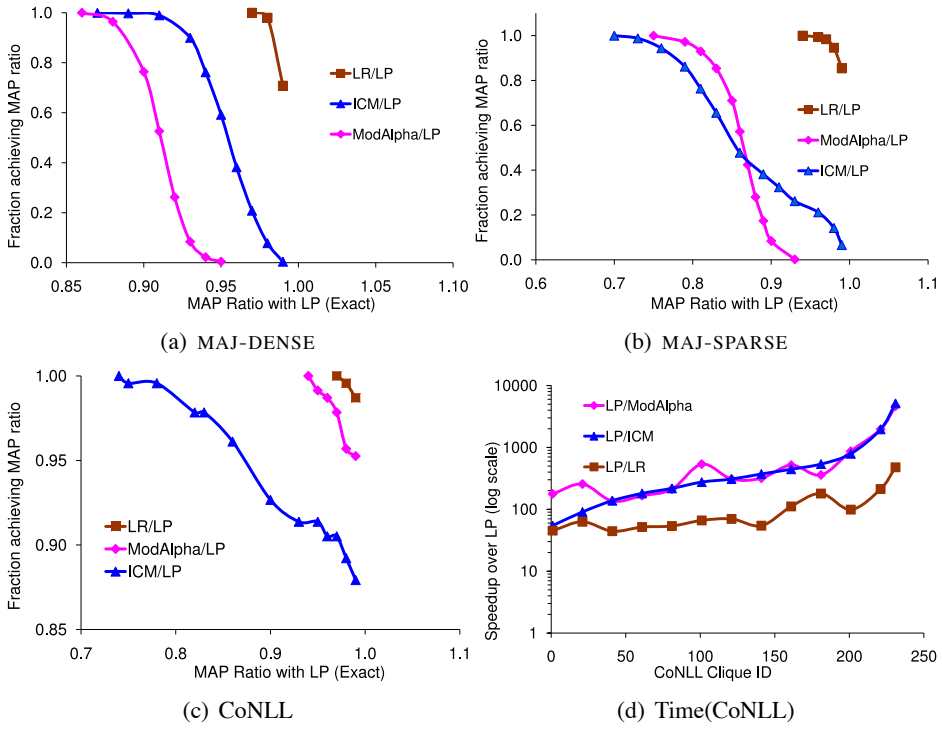


Figure 4: (a)-(c) MAP quality of modified α-pass, ICM, LR vs LP on MAJORITY potentials over MAJ-DENSE, MAJ-SPARSE, and CoNLL. To be interpreted in the same way as Figure 3. (d) Runtime ratios vs LP of these algorithms for each CoNLL clique.

only $3-5$ times more expensive than ICM and ModAlpha on average. Thus, for practical majority potentials, LR and ModAlpha quickly provide highly accurate solutions.

7.2 Collective Labeling of Repeated Words

We now establish that for collective inference setups like the one in Figure 1(d), message passing on the cluster graph (denoted CI) is a better option than the alternatives. This would justify the design of special clique inference algorithms such as α-pass and LR.

We consider information extraction over text records, and define cliques over repeated occurrences of words. We create two versions of the experiment—with Potts and MAJORITY potentials on the cliques respectively. Message computation at those cliques will be done using α-pass and LR respectively, as we have already established their efficiency and accuracy in Section 7.1.

For the edge-decomposable Potts version, we compare CI against TRW-S on the giant pairwise graph. Other algorithms such as FastPD, FastCut, and α-exp are not applicable as none of the chain edge potentials in the giant graph are semi-metric or submodular. For the MAJORITY version, we compare CI against the stacking approach of Krishnan and Manning (2006).

We report results on three data sets—the Address data set consisting of roughly 400 non-US postal addresses, the Cora data set (McCallum et al., 2000) containing 500 bibliographic records, and the CoNLL'03 data set. The training splits were 30%, 10% and 100% respectively for the three

data sets, and the parameter λ for Potts was set to 0.2,1 and 0.05 using cross-validation on a held out split. The MAJORITY parameter W was learnt generatively through label co-occurrence statistics in the cliques present in the training data. We report the token-F1 scores as a measure of accuracy of the various approaches.

Figure 5 reports the combined token-F1 over all labels except 'Other'. Unless specified otherwise, all the approaches post statistically significant gains over the base model. The accuracies show only modest improvements over the base model. This is because our cliques are of a highly limited form and so we cannot expect to correct too many errors using a collective model. We will look at more complex cliques in Section 7.3. Coming back to Figure 5, for MAJORITY potentials, CI is superior to the stacking based approach. The difference is statistically significant for Cora and CoNLL'03. For the Potts version, TRW-S and CI provide similar gains over Address and Cora. We could not run TRW-S on CoNLL'03, as the resulting graph was too big for the TRW-S code to handle.

We now compare the different approaches on running time. In Figure 6 we plot the accuracy of the two methods versus the number of iterations. CI achieves its best accuracy after just one round of message passing, whereas TRW-S takes around 20 iterations. In terms of clock time, an iteration of TRW-S costs $\sim$ 3.2s for CORA, and that of CI costs 3s, so CI is roughly an order of magnitude faster than TRW-S for the same accuracy levels. The comparison was similar for the Address data set.

Potential	**Model**	**Addr**	**Cora**	**CoNLL**
	Base	81.5	88.9	87.0
Potts	CI	81.9	89.7	88.8
	TRW-S	81.9	89.7	-
Majority	CI	82.2	89.6	88.8
	Stacking	81.7*	87.5$\downarrow$	87.8

Figure 5: Token-F1 of various collective inference schemes. F1 averaged over five splits for Address and Cora. '*' and $\downarrow$ denote statistically insignificant difference and significant loss over Base respectively.

7.3 Domain Adaptation

We move on to a generalization of our collective inference framework, and show that capturing associativity of a richer set of properties can help us in domain adaptation. We focus on a citation extraction task, where the aim is to adapt a sequential model across widely varying publications pages of authors. Our data set consists of 433 bibliographic entries from the web-pages of 31 authors, hand-labeled with 14 labels such as Title, Author, Venue, Location and Year. Bibliographic entries across different authors differ in many aspects like label-ordering, missing labels, punctuation, HTML formatting and bibliographic style.

A fraction of the 31 domains were used to train a baseline sequential model. The model was trained with the LARank algorithm of Bordes et al. (2007), using the BCE encoding for the labels.

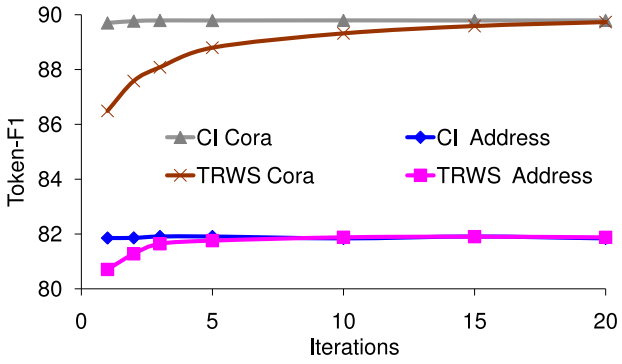


Figure 6: F1 vs iterations for CI vs TRW-S on Cora and Address.

We used standard extraction features in a window around each token, along with label transition features (Peng and McCallum, 2004).

For our collective framework, we used the following Markovian properties:

$$\begin{aligned} g_1(\mathbf{x},\mathbf{y}) &= \text{First non-Other label in } \mathbf{y} \\ g_2(\mathbf{x},\mathbf{y}) &= \text{Token before the Title segment in } \mathbf{y} \\ g_3(\mathbf{x},\mathbf{y}) &= \text{First non-Other label after Title in } \mathbf{y} \\ g_4(\mathbf{x},\mathbf{y}) &= \text{First non-Other label after Venue in } \mathbf{y} \end{aligned}$$

Inside a domain, any one of the above properties will predominantly favor one value, for example, g_3 might favor the value 'Author' in one domain, and 'Date' in another. Thus these properties encourage consistent labeling around the Title and Venue segments. We use Potts potential for each property, setting $\lambda = 1$ using cross-validation.

We reiterate that there are no alternative collective inference schemes for this property-based framework. This is primarily because other algorithms like TRW-S or ordinary belief propagation cannot deal with property-based separators (ref. Section 6).

The performance results of CI with the above properties versus the baseline model are presented in Figure 7. For the test domains, we report token-F1 of the important labels—Title, Author and Venue. The accuracies are averaged over five trials. CI leads to upto 25% reduction over the base test error for Venue and Title, labels for which we had defined related properties. The gain is statistically significant ($p < 0.05$). Though the improvement is more prominent when only a few domains are available for training, we continue to see an improvement even with more training domains as there invariably are new styles in the test data. Figure 8 shows the error reduction on individual test domains for one particular train-test split of five and 26 domains respectively. The errors are computed from the combined token F1 scores of Title, Venue and Author. For some domains the errors are reduced by more than 50%. Collective inference increases errors in only two domains. Such an increase happens when most of the records in the domain take on wrong property values, so collective inference ends up reinforcing those errors by wrongly biasing the remaining minority of the records that have correct property values.

Finally, we mention that for this task, applying the classical collective inference setup with cliques over word repetitions leads to very minor gains. This is because most of the word cliques

already agree on their vertex labels under the base model, so collective inference with word cliques does not add too much value. In this context, the generalized collective inference framework is indeed a much more accurate mechanism for joint labeling.

Train	**Title**		**Venue**		**Author**	
(%)	Base	CI	Base	CI	Base	CI
5	70.7	74.8	58.8	62.5	74.1	74.3
10	78.0	82.1	69.2	72.2	75.6	75.9
20	85.8	88.6	76.7	78.9	80.7	80.7
30	91.7	93.0	81.5	82.6	87.7	88.0
50	92.3	94.2	83.5	84.5	89.4	90.0

Figure 7: Token-F1 of CI and Base

8. Related Work

We group the known approaches into various categories and compare them with our collective inference framework.

8.1 Generic Collective Inference Approaches

Collective graphical models have been used to capture associativity in many text mining tasks such as IE, entity labeling, and document classification (Sutton and McCallum, 2004; Finkel et al., 2005; Bunescu and Mooney, 2004; Krishnan and Manning, 2006; Kulkarni et al., 2009; Chakrabarti et al., 1998; Lu and Getoor, 2003; Taskar et al., 2004). However these models use generic algorithms like ordinary belief propagation (Sutton and McCallum, 2004; Bunescu and Mooney, 2004; Taskar et al., 2004), Gibbs sampling (Finkel et al., 2005), local search (Lu and Getoor, 2003) or multi-stage schemes (Krishnan and Manning, 2006). Our framework is general enough to support various clique potentials, yet exploits the structure of the potential to efficiently compute a full set of messages for collective inference.

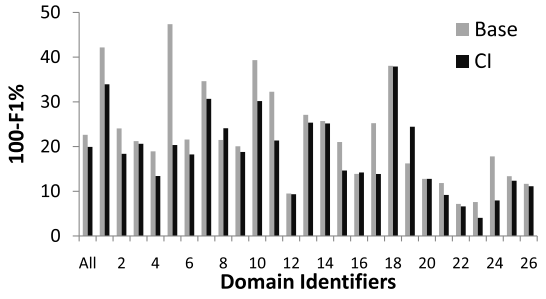


Figure 8: Per-domain F1-error

8.2 MAP Inference on Pairwise Models

We provide only a brief overview of the many recent advances in inference over pairwise graphs since our main interest lies in higher-order potentials. These approaches fall under two broad categories. The first category has message passing algorithms that solve the dual of an LP relaxation, including TRW-S (Kolmogorov, 2006), max-sum diffusion (Werner, 2009), and other convergent alternatives (Meltzer et al., 2009; Ravikumar et al., 2010). These LP relaxations, that only impose local pairwise constraints, have been tightened in various ways using cycle and higher order marginal constraints (Sontag et al., 2008; Werner, 2009; Komodakis and Paragios, 2008; Kumar et al., 2009). The second category includes combinatorial algorithms based on graph-cuts. For binary labels, two well known methods are the graph-cut method for submodular potentials (Boykov et al., 2001) and the Quadratic Pseudo-Boolean Optimization method (QPBO) for getting a partial solution for arbitrary potentials (Boros and Hammer, 2002; Kovtun, 2003; Kolmogorov and Rother, 2007). For multi-label models with metric edge potentials, the α-expansion algorithm of Boykov et al. (2001) provides a 1/2-approximation. α-expansion has subsequently been generalized, analyzed and optimized (Veksler, 2007; Kumar and Torr, 2008b; Lempitsky et al., 2007; Komodakis and Tziritas, 2005; Komodakis et al., 2007b; Alahari et al., 2008). In particular, the FastPD algorithm of Komodakis et al. (2007b) is a primal-dual generalization which works even with semi-metric edge potentials. However our chain edge potentials are not even semi-metric. Similarly the partial optimality guarantee of QPBO is of limited appeal, as our model has a large fraction of non-submodular edges.

8.3 Alternate Collective Inference Frameworks

Recently, two inference frameworks, the LP relaxation of Werner (2008) and the dual-decomposition framework of Komodakis and Paragios (2009) have been extended to handle higher-order potentials. In the LP relaxation framework, a max-sum diffusion algorithm is used to solve the dual and each step of the algorithm requires the computation of max-marginals from the higher-order cliques. In the dual-decomposition framework, the model is decomposed into tractable components and the component MAPs or marginals are used to construct a sub-gradient of the collective inference LP. Like cluster message passing, these frameworks also allow the plugging-in of arbitrary algorithms for computing max-marginals at each clique. Thus, our clique inference algorithms can be used unchanged in these two frameworks. However, while these alternative frameworks provide interesting new ways of looking at inference, they do not necessarily provide faster convergence guarantees. For example, as noted in Werner (2009), max-sum diffusion converges very slowly as compared to TRW-S even for binary potentials. We showed in Section 7.1 that dual decomposition is significantly slower than α-pass when used for clique inference.

8.4 Transforming Higher-Order Cliques to Pairwise Potentials

A lot of recent work on higher-order clique potentials in vision deals with reducing these potentials to pairwise and then applying α-expansion or QPBO as applicable (Ishikawa, 2009; Kohli et al., 2008; Ramalingam et al., 2008; Rother et al., 2009; Kohli and Kumar, 2010). These transformations add auxiliary nodes, which are few in number when the potentials are truncated and sparse (Rother et al., 2009). However our potentials are dense and not truncated. Consequently these techniques will introduce too many extra nodes, for example, exponential in n (Ishikawa, 2009), or $O(mn)$

(Kohli et al., 2008). Similarly, the method of Kohli and Kumar (2010) will expand the node set by a factor equal to the number of functions used to approximate our potential (and this can be quite large). Too many auxiliary nodes makes inference using α-expansion/QPBO quite expensive. In contrast we perform clique inference without any transformation, and our clique inference bounds are better than the alternatives.

We also note that there are transformations that extend α-expansion to metric potentials over triangles (Kolmogorov and Zabih, 2004), and truncated versions of Potts potentials over arbitrarily large cliques (Kohli et al., 2007, 2009). Our symmetric potentials are not truncated so the same transformations cannot be applied to our model.

8.5 Special Higher-Order Potentials

An orthogonal body of work deals with special higher-order potentials that allow efficient message computation, such as linear-constrained potentials (Potetz and Lee, 2008), convex and order-based potentials (Tarlow et al., 2010), and decomposable potentials (Chen et al., 2008). Although our cluster graph can seamlessly incorporate these potentials and their clique inference algorithms, it is fruitful to compare some of these potentials with ours. Linear-constrained potentials assume that the labels are numeric and the potential's value depends only on a linear combination of node labels. Such potentials allow $O(nm^2)$ message computation using clever variable substitution during belief propagation. When we have only two labels, our symmetric potentials can be encoded as linear-constrained potentials, thus leading to $O(n)$ clique inference as compared to $O(n \log n)$ using α-pass. However, this encoding does not have an inexpensive generalization to *discrete* multi-label cliques. Also, representing multi-label potentials using a collection of linear-constrained potentials can make clique inference exponential in m (Potetz and Lee, 2008). Chen et al. (2008) propose *decomposable potentials* that are sums of a few sub-potentials, each of which is a product of low-arity functions. This includes the *Voting Potential*, which combines associativity scores additively along edges unlike multiplicatively as in Potts.[3] For some decomposable potentials, including the Voting Potential, exact messages can be computed in closed form.

Symmetric potentials have been used in special ways in other inference tasks too. Jaimovich et al. (2007) proposed a framework for inference over relational data where symmetric potentials are used to collapse large instance-level factor graphs into template-level graphs. Similarly, Milch et al. (2008) proposes lifting techniques with counting potentials. In our framework we have a mix of symmetric potentials, individual node and non-symmetric edge potentials. We therefore cannot perform any kind of collapsing or lifted inference.

9. Conclusions and Future Work

We presented a general collective inference framework that exploits the associativity of a rich set of properties of instances and their labelings. We argued that cluster message passing, which exploits the special associative structure and computes a complete set of messages, is a more principled inference mechanism than other cluster-oblivious or message-oblivious approaches. We demonstrated the effectiveness of the framework on a real-life domain adaptation task.

3. Although Potts decomposes additively along the clique edges, the probability $P(\mathbf{y}|\mathbf{x})$ is exponential in the clique potential term, which makes it multiplicative.

We presented potential-specific combinatorial algorithms for message computation in associative cliques. We presented the α-pass algorithm which is sub-quadratic in the clique size and gives the exact MAP for all MAX clique potentials, and any symmetric potential with two labels, and a tight approximation guarantee of $\frac{13}{15}$ on the Potts potential. We showed that α-pass is significantly faster while providing the same or better accuracy than alternatives such as TRW-S, graph-cuts, and their recently proposed improved versions. We gave a Lagrangian relaxation method for generating messages from a clique with majority potential. This algorithm is at least two orders of magnitude faster than an exact algorithm and more accurate than other approximate approaches.

Our future work includes automated property induction to figure out rich associative properties in unlabeled domains. We are also interested in applying symmetric potentials to general dense subgraphs instead of cliques. We believe that this might help in semi-supervised learning tasks.

Appendix A. Proofs

Theorem 7 *$F(\hat{\mathbf{y}}) \geq \frac{13}{15} F(\mathbf{y}^*)$. Further, this ratio is tight.*

Proof The proof is by contradiction. Suppose there is an instance where $F(\hat{\mathbf{y}}) < \frac{13}{15} F(\mathbf{y}^*)$. Wlog assume that $\lambda = 1$ and $n_1 \geq n_2 \geq \ldots \geq n_k > 0,\ (2 \leq k \leq m)$ be the non-zero counts in the optimal solution and let ψ^* be its vertex score. Thus $F(\mathbf{y}^*) = \psi^* + n_1^2 + n_2^2 + \ldots + n_k^2$.

Now, $F(\hat{\mathbf{y}})$ is at least $\psi^* + n_1^2$ (ref. Claim 5.1). This implies $\frac{\psi^* + n_1^2}{\psi^* + n_1^2 + \ldots + n_k^2} \leq \frac{13}{15}$, that is, $2(\psi^* + n_1^2) < 13(n_2^2 + \ldots + n_k^2)$ or

$$\psi^* < \frac{13}{2}(n_2^2 + \ldots + n_k^2) - n_1^2. \tag{18}$$

Since k labels have non-zero counts, and the vertex score is ψ^*, at least ψ^*/k of the vertex score is assigned to one label. Considering a solution where all vertices are assigned to this label, we get $F(\hat{\mathbf{y}}) \geq \psi^*/k + n^2$.

Therefore $F(\mathbf{y}^*) > 15/13(n^2 + \psi^*/k)$.

Since $F(\mathbf{y}^*) = \psi^* + n_1^2 + \ldots + n_k^2$, we get:

$$\psi^* > \frac{15kn^2 - 13k(n_1^2 + \ldots + n_k^2)}{13k - 15}. \tag{19}$$

We show that Equations 18 and 19 contradict each other. It is sufficient to show that for all $n_1 \geq \ldots \geq n_k \geq 1$,

$$\frac{15kn^2 - 13k(n_1^2 + \ldots n_k^2)}{13k - 15} \geq \frac{13}{2}(n_2^2 + \ldots + n_k^2) - n_1^2.$$

Simplifying, this is equivalent to

$$kn^2 - \frac{13}{2}(k-1)(n_2^2 + \ldots + n_k^2) - n_1^2 \geq 0.$$

Consider a sequence $n_1, \ldots, n_k$ for which the expression on the left hand side is minimized. If $n_i > n_{i+1}$ then we must have $n_l = 1\ \forall l \geq i+2$. Otherwise, replace n_{i+1} by $n_{i+1} + 1$ and decrement n_j by 1, where j is the largest index for which $n_j > 1$. This gives a new sequence for which the value of the expression is smaller. Therefore the sequence must be of the form $n_i = n_1$ for $1 \leq i < l$ and $n_i = 1$ for $i > l$, for some $l \geq 2$. Further, considering the expression as a function of n_l, it is

quadratic with a negative second derivative. So the minimum occurs at one of the extreme values $n_l = 1$ or $n_l = n_1$. Therefore we only need to consider sequences of the form $n_1, \ldots, n_1, 1, \ldots, 1$ and show that the expression is non-negative for these.

In such sequences, differentiating with respect to n_1, the derivative is positive for $n_1 \geq 1$, which means that the expression is minimized for the sequence $1, \ldots, 1$. Now it is easy to verify that it is true for such sequences. The expression is zero only for the sequence $1, 1, 1$, which gives the worst case example.

We now illustrate the tightness of this ratio through a pathological instance where the solution of α-pass is exactly $\frac{13}{15}$ of the optimal. The clique is constructed as follows. Let $m = n + 3$ and $\lambda = 1$. For the first $n/3$ vertices let $\psi_{u1} = 4n/3$, for the next $n/3$ vertices let $\psi_{u2} = 4n/3$, and for the remaining $n/3$ let $\psi_{u3} = 4n/3$. Also for all vertices let $\psi_{u(u+3)} = 4n/3$. All other vertex scores are zero. The optimal solution is to assign the first three labels $n/3$ vertices each, yielding a score of $4n^2/3 + 3(\frac{n}{3})^2 = 5n^2/3$. The first α-pass with $\alpha = 1$, where initially a vertex u is assigned its vertex optimal label $u + 3$, will assign the first $n/3$ vertices label 1. This keeps the sum of total vertex scores unchanged at $4n^2/3$, the clique score increases to $n^2/9 + 2n/3$ and total score = $4n^2/3 + n^2/9 + 2n/3 = 13n^2/9 + 2n/3$. No subsequent combinations with any other label α can improve this score. Thus, the score of α-pass is $\frac{13}{15}$ of the optimal in the limit $n \to \infty$. ■

Theorem 8 *Generalized α-pass enjoys an approximation bound of $\frac{8}{9}$, that is, $F(\hat{\mathbf{y}}) \geq \frac{8}{9} F(\mathbf{y}^*)$.*

Proof This bound is achieved if we run the algorithm with $q = 2$. Let the optimal solution have counts $n_1 \geq n_2 \geq \ldots \geq n_m$ and let its vertex score be ψ^*. For simplicity let $a = n_1/n$, $b = n_2/n$ and $c = \psi^*/n^2$. Then $F(\mathbf{y}^*)/n^2 \leq c + a^2 + b(1-a)$, $F(\hat{\mathbf{y}})/n^2 \geq c + a^2$ and $F(\hat{\mathbf{y}})/n^2 \geq c + \frac{(a+b)^2}{2}$.
Case 1: $a^2 \geq \frac{(a+b)^2}{2}$. Then $F(\mathbf{y}^*) - F(\hat{\mathbf{y}}) \leq bn^2(1-a)$. For a given value of a, this is maximized when b is as large as possible. For Case 1 to hold, the largest possible value of b is given by $a^2 = \frac{(a+b)^2}{2}$, which gives $b = a(\sqrt{2}-1)$. Therefore $F(\mathbf{y}^*) - F(\hat{\mathbf{y}}) \leq \frac{n^2(\sqrt{2}-1)}{4} < \frac{n^2}{8} \leq \frac{F(\hat{\mathbf{y}})}{8}$, that is, $F(\hat{\mathbf{y}}) \geq \frac{8}{9} F(\mathbf{y}^*)$.
Case 2: $a^2 \leq \frac{(a+b)^2}{2}$. This holds if $b \geq (\sqrt{2}-1)a$. Since $a + b \leq 1$, this is possible only if $a \leq 1/\sqrt{2}$. Now $\frac{F(\mathbf{y}^*) - F(\hat{\mathbf{y}})}{n^2} \leq a^2 + b(1-a) - (a+b)^2/2 = \frac{a^2 - 4ab + 2b - b^2}{2}$.

For a given a, this expression is quadratic in b with a negative second derivative. This is maximized (by differentiating) for $b = 1 - 2a$. Since $b \leq a$, this value is possible only if $a \geq 1/3$. Similarly, for case 2 to hold with this value of b, we must have $a \leq \sqrt{2} - 1$. Substituting this value of b, the difference in scores is $\frac{5a^2 - 4a + 1}{2}$.

Since this is quadratic with a positive second derivative, it is maximized when a has either the minimum or maximum possible value. For $a = 1/3$ this value is $1/9$, while for $a = \sqrt{2} - 1$, it is $10 - 7\sqrt{2}$. In both cases, it is less than $1/8$.

If $a \leq 1/3$ the maximum is achieved when $b = a$. In this case, the score difference is at most $(a - 2a^2)$ which is maximized for $a = 1/4$, where the value is $1/8$. (This is the worst case).

For $\sqrt{2} - 1 < a \leq 1/\sqrt{2}$, the maximum will occur for $b = (\sqrt{2}-1)a$. Substituting this value for b, the score difference is $(\sqrt{2}-1)(a - a^2)$, which is maximized for $a = 1/2$, where its value is $(\sqrt{2}-1)/4 < 1/8$. ■

References

Karteek Alahari, Pushmeet Kohli, and Philip H. S. Torr. Reduce, reuse & recycle: efficiently solving multi-label mrfs. In *CVPR*, 2008.

David Blei, Drew Bagnell, and Andrew McCallum. Learning with scope, with application to information extraction and classification. In *UAI*, 2002.

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, 2006.

Antoine Bordes, Léon Bottou, Patrick Gallinari, and Jason Weston. Solving multiclass support vector machines with larank. In *ICML*, pages 89–96, 2007.

Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, volume 26, no. 9, pages 1124–1137, 2004.

Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

Razvan Bunescu and Raymond J. Mooney. Collective information extraction with relational markov networks. In *ACL*, 2004.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM*, 2010.

Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. *SIGMOD Rec.*, 27(2):307–318, 1998.

Shann-Ching Chen, Geoffrey J. Gordon, and Robert F. Murphy. Graphical models for structured classification, with an application to interpreting images of protein subcellular location patterns. *JMLR*, 9:651–682, 2008. ISSN 1533-7928.

John Duchi, Daniel Tarlow, Gal Elidan, and Daphne Koller. Using combinatorial optimization within max-product belief propagation. In *Advances in Neural Information Processing Systems (NIPS 2006)*, 2007.

Hazem Elmeleegy, Jayant Madhavan, and Alon Halevy. Harvesting relational tables from lists on the web. In *VLDB*, 2009.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

Rahul Gupta and Sunita Sarawagi. Answering table augmentation queries from unstructured lists on the web. In *PVLDB*, 2009.

Rahul Gupta, Ajit A. Diwan, and Sunita Sarawagi. Efficient inference with cardinality-based clique potentials. In *Proceedings of the 24th International Conference on Machine Learning (ICML), USA*, 2007.

Rahul Gupta, Sunita Sarawagi, and Ajit A. Diwan. Generalized collective inference with symmetric clique potentials, 2009. URL `http://arxiv.org/abs/0907.0589v2`.

Hiroshi Ishikawa. Higher-order clique reduction in binary graph cut. In *CVPR*, pages 2993–3000, 2009.

Ariel Jaimovich, Ofer Meshi, and Nir Friedman. Template based inference in symmetric relational markov random fields. In *UAI*, 2007.

Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *J. ACM*, 49(5):616–639, 2002. doi: http://doi.acm.org/10.1145/585265.585268.

Pushmeet Kohli and M. Pawan Kumar. Energy minimization for linear envelope mrfs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Pushmeet Kohli, M. Pawan Kumar, and Philip H. S. Torr. P3 & beyond: Solving energies with higher order cliques. In *CVPR*, 2007.

Pushmeet Kohli, Alexander Shekhovtsov, Carsten Rother, Vladimir Kolmogorov, and Philip H. S. Torr. On partial optimality in multi-label mrfs. In *ICML*, pages 480–487, 2008.

Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006. ISSN 0162-8828.

Vladimir Kolmogorov and Carsten Rother. Minimizing nonsubmodular functions with graph cuts – a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1274–1279, 2007.

Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.

Nikos Komodakis and Nikos Paragios. Beyond loose lp-relaxations: optimizing mrfs by repairing cycles. In *ECCV*, 2008.

Nikos Komodakis and Nikos Paragios. Beyond pairwise energies: efficient optimization for higher-order mrfs. In *CVPR*, 2009.

Nikos Komodakis and Georgios Tziritas. A new framework for approximate labeling via graph cuts. In *ICCV*, pages 1018–1025, 2005.

Nikos Komodakis and Georgios Tziritas. Approximate labeling via graph cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1436–1453, 2007.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. Mrf optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007a.

Nikos Komodakis, Georgios Tziritas, and Nikos Paragios. Fast, approximately optimal solutions for single and dynamic mrfs. In *CVPR*, 2007b.

Nikos Komodakis, Georgios Tziritas, and Nikos Paragios. Performance vs computational efficiency for optimizing single and dynamic mrfs: Setting the state of the art with primal-dual strategies. *Comput. Vis. Image Underst.*, 112(1):14–29, 2008.

Ivan Kovtun. Partial optimal labeling search for a np-hard subclass of (max, +) problems. In *DAGM-Symposium*, pages 402–409, 2003.

Vijay Krishnan and Christopher D. Manning. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *ACL-COLING*, 2006.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *SIGKDD*, 2009.

M. Pawan Kumar and Philip H. S. Torr. Efficiently solving convex relaxations for map estimation. In *ICML*, 2008a.

M. Pawan Kumar and Philip H. S. Torr. Improved moves for truncated convex models. In *NIPS*, pages 889–896, 2008b.

M. Pawan Kumar, Vladimir Kolmogorov, and Philip H. S. Torr. An analysis of convex relaxations for map estimation of discrete mrfs. *JMLR*, 10:71–106, 2009.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML-2001)*, Williams, MA, 2001.

Victor S. Lempitsky, Carsten Rother, and Andrew Blake. Logcut - efficient graph cut optimization for markov random fields. In *ICCV*, pages 1–8, 2007.

Qing Lu and Lise Getoor. Link-based classification. In *Machine Learning, ICML*, pages 496–503, 2003.

Gideon Mann and Andrew McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *ICML*, 2007.

Andrew McCallum, Kamal Nigam, Jason Reed, Jason Rennie, and Kristie Seymore. Cora: computer science research paper search engine. http://cora.whizbang.com/, 2000.

Talya Meltzer, Amir Globerson, and Yair Weiss. Convergent message passing algorithms - a unifying view. In *UAI*, 2009.

Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted probabilistic inference with counting formulas. In *AAAI*, 2008.

Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, chapter 11, pages 247–254. Prentice Hall, Englewood Cliffs, NJ., 1982.

Fuchun Peng and Andrew McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL*, pages 329–336, 2004.

Brian Potetz and Tai Sing Lee. Efficient belief propagation for higher-order cliques using linear constraint nodes. *Computer Vision and Image Understanding*, 112(1):39–54, 2008.

Srikumar Ramalingam, Pushmeet Kohli, Karteek Alahari, and Philip H. S. Torr. Exact inference in multi-label crfs with higher order cliques. In *CVPR*, 2008.

Pradeep Ravikumar, Alekh Agarwal, and Martin J. Wainwright. Message-passing for graph-structured linear programs: proximal methods and rounding schemes. *Journal of Machine Learning Research*, 11:1043–1080, 2010.

Carsten Rother, Pushmeet Kohli, Wei Feng, and Jiaya Jia. Minimizing sparse higher order energy functions of discrete variables. In *CVPR*, 2009.

David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. Tightening lp relaxations for map using message passing. In *UAI*, 2008.

Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, 2004.

Rick Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields. In *ECCV 2006*, volume 2, pages 16–29, 2006.

Daniel Tarlow, Inmar Givoni, and Richard Zemel. Hop-map: efficient message passing with high order potentials. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AI-STATS)*, volume 9, pages 812–819. JMLR: W&CP, 2010.

Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning associative markov networks. In *Twenty First International Conference on Machine Learning (ICML04)*, 2004.

Olga Veksler. Graph cut based optimization for mrfs with truncated convex priors. In *CVPR*, 2007.

Tomás Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (map-mrf). In *CVPR*, 2008.

Tomas Werner. Revisiting the linear programming relaxation approach to gibbs energy minimization and weighted constraint satisfaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2009. ISSN 0162-8828.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring Artificial Intelligence in the New Millenium*, 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 1-55860-811-7.

Journal of Machine Learning Research 11 (2010) 3137-3181 Submitted 1/10; Revised 7/10; Published 11/10

A Generalized Path Integral Control Approach to Reinforcement Learning

Evangelos A.Theodorou ETHEODOR@USC.EDU
Jonas Buchli JONAS@BUCHLI.ORG
Stefan Schaal* SSCHAAL@USC.EDU
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-2905, USA

Editor: Daniel Lee

Abstract

With the goal to generate more scalable algorithms with higher efficiency and fewer open parameters, reinforcement learning (RL) has recently moved towards combining classical techniques from optimal control and dynamic programming with modern learning techniques from statistical estimation theory. In this vein, this paper suggests to use the framework of stochastic optimal control with path integrals to derive a novel approach to RL with parameterized policies. While solidly grounded in value function estimation and optimal control based on the stochastic Hamilton-Jacobi-Bellman (HJB) equations, policy improvements can be transformed into an approximation problem of a path integral which has no open algorithmic parameters other than the exploration noise. The resulting algorithm can be conceived of as model-based, semi-model-based, or even model free, depending on how the learning problem is structured. The update equations have no danger of numerical instabilities as neither matrix inversions nor gradient learning rates are required. Our new algorithm demonstrates interesting similarities with previous RL research in the framework of probability matching and provides intuition why the slightly heuristically motivated probability matching approach can actually perform well. Empirical evaluations demonstrate significant performance improvements over gradient-based policy learning and scalability to high-dimensional control problems. Finally, a learning experiment on a simulated 12 degree-of-freedom robot dog illustrates the functionality of our algorithm in a complex robot learning scenario. We believe that **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}) offers currently one of the most efficient, numerically robust, and easy to implement algorithms for RL based on trajectory roll-outs.

Keywords: stochastic optimal control, reinforcement learning, parameterized policies

1. Introduction

While reinforcement learning (RL) is among the most general frameworks of learning control to create truly autonomous learning systems, its scalability to high-dimensional continuous state-action systems, for example, humanoid robots, remains problematic. Classical value-function based methods with function approximation offer one possible approach, but function approximation under the non-stationary iterative learning process of the value-function remains difficult when one exceeds about 5-10 dimensions. Alternatively, direct policy learning from trajectory roll-outs has recently made significant progress (Peters, 2007), but can still become numerically brittle and full of open

∗. Also at ATR Computational Neuroscience Laboratories, Kyoto 619-0288, Japan.

tuning parameters in complex learning problems. In new developments, RL researchers have started to combine the well-developed methods from statistical learning and empirical inference with classical RL approaches in order to minimize tuning parameters and numerical problems, such that ultimately more efficient algorithms can be developed that scale to significantly more complex learning system (Dayan and Hinton, 1997; Koeber and Peters, 2008; Peters and Schaal, 2008c; Toussaint and Storkey, 2006; Ghavamzadeh and Yaakov, 2007; Deisenroth et al., 2009; Vlassis et al., 2009; Jetchev and Toussaint, 2009).

In the spirit of these latter ideas, this paper addresses a new method of probabilistic reinforcement learning derived from the framework of stochastic optimal control and path integrals, based on the original work of Kappen (2007) and Broek et al. (2008). As will be detailed in the sections below, this approach makes an appealing theoretical connection between value function approximation using the stochastic HJB equations and direct policy learning by approximating a path integral, that is, by solving a statistical inference problem from sample roll-outs. The resulting algorithm, called **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}), takes on a surprisingly simple form, has no open algorithmic tuning parameters besides the exploration noise, and it has numerically robust performance in high dimensional learning problems. It also makes an interesting connection to previous work on RL based on probability matching (Dayan and Hinton, 1997; Peters and Schaal, 2008c; Koeber and Peters, 2008) and motivates why probability matching algorithms can be successful.

This paper is structured into several major sections:

- Section 2 addresses the theoretical development of stochastic optimal control with path integrals. This is a fairly theoretical section. For a quick reading, we would recommend Section 2.1 for our basic notation, and Table 1 for the final results. Exposing the reader to a sketch of the details of the derivations opens the possibility to derive path integral optimal control solutions for other dynamical systems than the one we address in Section 2.1.

  The main steps of the theoretical development include:

  - Problem formulation of stochastic optimal control with the stochastic Hamilton-Jacobi-Bellman (HJB) equation
  - The transformation of the HJB into a linear PDE
  - The generalized path integral formulation for control systems with controlled and uncontrolled differential equations
  - General derivation of optimal controls for the path integral formalism
  - Path integral optimal control applied to special cases of control systems

- Section 3 relates path integral optimal control to reinforcement learning. Several main issues are addressed:

  - Reinforcement learning with parameterized policies
  - Dynamic Movement Primitives (DMP) as a special case of parameterized policies, which matches the problem formulation of path integral optimal control.
  - Derivation of **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}), which is an application of path integral optimal control to DMPs.

- Section 4 discusses related work.

- Section 5 illustrates several applications of $\mathbf{PI}^2$ to control problems in robotics.
- Section 6 addresses several important issues and characteristics of RL with $\mathbf{PI}^2$.

2. Stochastic Optimal Control with Path Integrals

The goal in stochastic optimal control framework is to control a stochastic dynamical system while minimizing a performance criterion. Therefore, stochastic optimal control can be thought as a constrained optimization problem in which the constrains corresponds to stochastic dynamical systems. The analysis and derivations of stochastic optimal control and path integrals in the next sections rely on the Bellman Principle of optimality (Bellman and Kalaba, 1964) and the HJB equation.

2.1 Stochastic Optimal Control Definition and Notation

For our technical developments, we will use largely a control theoretic notation from trajectory-based optimal control, however, with an attempt to have as much overlap as possible with the standard RL notation (Sutton and Barto, 1998). Let us define a finite horizon cost function for a trajectory τ_i (which can also be a piece of a trajectory) starting at time t_i in state $\mathbf{x}_{t_i}$ and ending at time[1] t_N

$$R(\tau_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t \, dt, \tag{1}$$

with $\phi_{t_N} = \phi(x_{t_N})$ denoting a terminal reward at time t_N and r_t denoting the immediate cost at time t. In stochastic optimal control (Stengel, 1994), the goal is to find the controls $\mathbf{u}_t$ that minimize the value function:

$$V(\mathbf{x}_{t_i}) = V_{t_i} = \min_{\mathbf{u}_{t_i:t_N}} E_{\tau_i}\left[R(\tau_i)\right], \tag{2}$$

where the expectation $E_{\tau_i}[.]$ is taken over all trajectories starting at $\mathbf{x}_{t_i}$. We consider the rather general class of control systems:

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t,t) + \mathbf{G}(\mathbf{x}_t)\left(\mathbf{u}_t + \varepsilon_t\right) = \mathbf{f}_t + \mathbf{G}_t\left(\mathbf{u}_t + \varepsilon_t\right), \tag{3}$$

with $\mathbf{x}_t \in \Re^{n\times 1}$ denoting the state of the system, $\mathbf{G}_t = \mathbf{G}(\mathbf{x}_t) \in \Re^{n\times p}$ the control matrix, $\mathbf{f}_t = \mathbf{f}(\mathbf{x}_t) \in \Re^{n\times 1}$ the passive dynamics, $\mathbf{u}_t \in \Re^{p\times 1}$ the control vector and $\varepsilon_t \in \Re^{p\times 1}$ Gaussian noise with variance Σ_ε. As immediate cost we consider

$$r_t = r(\mathbf{x}_t,\mathbf{u}_t,t) = q_t + \frac{1}{2}\mathbf{u}_t^T\mathbf{R}\mathbf{u}_t, \tag{4}$$

where $q_t = q(\mathbf{x}_t,t)$ is an arbitrary state-dependent cost function, and $\mathbf{R}$ is the positive semi-definite weight matrix of the quadratic control cost. The stochastic HJB equation (Stengel, 1994; Fleming and Soner, 2006) associated with this stochastic optimal control problem is expressed as follows:

$$-\partial_t V_t = \min_{\mathbf{u}}\left(r_t + (\nabla_{\mathbf{x}}V_t)^T\mathbf{F}_t + \frac{1}{2}trace\left((\nabla_{\mathbf{xx}}V_t)\mathbf{G}_t\Sigma_\varepsilon\mathbf{G}_t^T\right)\right), \tag{5}$$

1. If we need to emphasize a particular time, we denote it by t_i, which also simplifies a transition to discrete time notation later. We use t without subscript when no emphasis is needed when this "time slice" occurs, t_0 for the start of a trajectory, and t_N for the end of a trajectory.

where $\mathbf{F}_t$ is defined as $\mathbf{F}_t = \mathbf{f}(\mathbf{x}_t, t) + \mathbf{G}(\mathbf{x}_t)\mathbf{u}_t$. To find the minimum, the cost function (4) is inserted into (5) and the gradient of the expression inside the parenthesis is taken with respect to controls $\mathbf{u}$ and set to zero. The corresponding optimal control is given by the equation:

$$\mathbf{u}(\mathbf{x}_t) = \mathbf{u}_t = -\mathbf{R}^{-1}\mathbf{G}_t^T(\nabla_{x_t} V_t).$$

Substitution of the optimal control above, into the stochastic HJB (5), results in the following nonlinear and second order Partial Differential Equation (PDE):

$$-\partial_t V_t = q_t + (\nabla_{\mathbf{x}} V_t)^T \mathbf{f}_t - \frac{1}{2}(\nabla_{\mathbf{x}} V_t)^T \mathbf{G}_t \mathbf{R}^{-1} \mathbf{G}_t^T (\nabla_{\mathbf{x}} V_t) + \frac{1}{2} trace\left((\nabla_{\mathbf{xx}} V_t)\mathbf{G}_t \Sigma_{\varepsilon} \mathbf{G}_t^T\right).$$

The $\nabla_{\mathbf{x}}$ and $\nabla_{\mathbf{xx}}$ symbols refer to the Jacobian and Hessian, respectively, of the value function with respect to the state $\mathbf{x}$, while ∂_t is the partial derivative with respect to time. For notational compactness, we will mostly use subscripted symbols to denote time and state dependencies, as introduced in the equations above.

2.2 Transformation of HJB into a Linear PDE

In order to find a solution to the PDE above, we use a exponential transformation of the value function:

$$V_t = -\lambda \log \Psi_t.$$

Given this logarithmic transformation, the partial derivatives of the value function with respect to time and state are expressed as follows:

$$\partial_t V_t = -\lambda \frac{1}{\Psi_t} \partial_t \Psi_t,$$

$$\nabla_{\mathbf{x}} V_t = -\lambda \frac{1}{\Psi_t} \nabla_{\mathbf{x}} \Psi_t,$$

$$\nabla_{\mathbf{xx}} V_t = \lambda \frac{1}{\Psi_t^2} \nabla_{\mathbf{x}} \Psi_t \, \nabla_{\mathbf{x}} \Psi_t^T - \lambda \frac{1}{\Psi_t} \nabla_{\mathbf{xx}} \Psi_t.$$

Inserting the logarithmic transformation and the derivatives of the value function we obtain:

$$\frac{\lambda}{\Psi_t} \partial_t \Psi_t = q_t - \frac{\lambda}{\Psi_t} (\nabla_{\mathbf{x}} \Psi_t)^T \mathbf{f}_t - \underline{\frac{\lambda^2}{2\Psi_t^2} (\nabla_{\mathbf{x}} \Psi_t)^T \mathbf{G}_t \mathbf{R}^{-1} \mathbf{G}_t^T (\nabla_{\mathbf{x}} \Psi_t)} + \frac{1}{2} trace(\Gamma), \tag{6}$$

where the term Γ is expressed as:

$$\Gamma = \left(\lambda \frac{1}{\Psi_t^2} \nabla_{\mathbf{x}} \Psi_t \, \nabla_{\mathbf{x}} \Psi_t^T - \lambda \frac{1}{\Psi_t} \nabla_{\mathbf{xx}} \Psi_t \right) \mathbf{G}_t \Sigma_{\varepsilon} \mathbf{G}_t^T.$$

The trace of Γ is therefore:

$$trace(\Gamma) = \underline{\lambda \frac{1}{\Psi^2} trace\left(\nabla_{\mathbf{x}} \Psi_t^T \mathbf{G}_t \Sigma_{\varepsilon} \mathbf{G}_t \nabla_{\mathbf{x}} \Psi_t\right)} - \lambda \frac{1}{\Psi_t} trace\left(\nabla_{\mathbf{xx}} \Psi_t \mathbf{G}_t \Sigma_{\varepsilon} \mathbf{G}_t^T\right). \tag{7}$$

Comparing the underlined terms in (6) and (7), one can recognize that these terms will cancel under the assumption of $\lambda \mathbf{R}^{-1} = \Sigma_\varepsilon$, which implies the simplification:

$$\lambda \mathbf{G}_t \mathbf{R}^{-1} \mathbf{G}_t^T = \mathbf{G}_t \Sigma_\varepsilon \mathbf{G}_t^T = \Sigma(\mathbf{x}_t) = \Sigma_t. \tag{8}$$

The intuition behind this assumption (cf. also Kappen, 2007; Broek et al., 2008) is that, since the weight control matrix $\mathbf{R}$ is inverse proportional to the variance of the noise, a high variance control input implies cheap control cost, while small variance control inputs have high control cost. From a control theoretic stand point such a relationship makes sense due to the fact that under a large disturbance (= high variance) significant control authority is required to bring the system back to a desirable state. This control authority can be achieved with corresponding low control cost in $\mathbf{R}$.

With this simplification, (6) reduces to the following form

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + \mathbf{f}_t^T (\nabla_{\mathbf{x}} \Psi_t) + \frac{1}{2} trace \left((\nabla_{\mathbf{xx}} \Psi_t) \mathbf{G}_t \Sigma_\varepsilon \mathbf{G}_t^T \right), \tag{9}$$

with boundary condition: $\Psi_{t_N} = \exp\left(-\frac{1}{\lambda}\phi_{t_N}\right)$. The partial differential equation (PDE) in (9) corresponds to the so called Chapman Kolmogorov PDE, which is of second order and linear. Analytical solutions of (9) cannot be found in general for general nonlinear systems and cost functions. However, there is a connection between solutions of PDEs and their representation as stochastic differential equation (SDEs), that is mathematically expressed by the Feynman-Kac formula (Øksendal, 2003; Yong, 1997). The Feynman-Kac formula (see appendix B) can be used to find distributions of random processes which solve certain SDEs as well as to propose numerical methods for solving certain PDEs. Applying the Feynman-Kac theorem, the solution of (9) is:

$$\Psi_{t_i} = E_{\boldsymbol{\tau}_i} \left(\Psi_{t_N} e^{-\int_{t_i}^{t_N} \frac{1}{\lambda} q_t dt} \right) = E_{\boldsymbol{\tau}_i} \left[\exp\left(-\frac{1}{\lambda}\phi_{t_N} - \frac{1}{\lambda} \int_{t_i}^{t_N} q_t \, dt \right) \right]. \tag{10}$$

Thus, we have transformed our stochastic optimal control problem into the approximation problem of a path integral. With a view towards a discrete time approximation, which will be needed for numerical implementations, the solution (10) can be formulated as:

$$\Psi_{t_i} = \lim_{dt \to 0} \int p(\boldsymbol{\tau}_i | \mathbf{x}_i) \exp\left[-\frac{1}{\lambda} \left(\phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt \right) \right] d\boldsymbol{\tau}_i, \tag{11}$$

where $\boldsymbol{\tau}_i = (\mathbf{x}_{t_i}, ..., \mathbf{x}_{t_N})$ is a sample path (or trajectory piece) starting at state $\mathbf{x}_{t_i}$ and the term $p(\boldsymbol{\tau}_i | \mathbf{x}_i)$ is the probability of sample path $\boldsymbol{\tau}_i$ conditioned on the start state $\mathbf{x}_{t_i}$. Since Equation (11) provides the exponential cost to go Ψ_{t_i} in state $\mathbf{x}_{t_i}$, the integration above is taken with respect to sample paths $\boldsymbol{\tau}_i = (\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+1}}, ..., \mathbf{x}_{t_N})$. The differential term $d\boldsymbol{\tau}_i$ is defined as $d\boldsymbol{\tau}_i = (d\mathbf{x}_{t_i}, ..., d\mathbf{x}_{t_N})$. Evaluation of the stochastic integral in (11) requires the specification of $p(\boldsymbol{\tau}_i | \mathbf{x}_i)$, which is the topic of our analysis in the next section.

2.3 Generalized Path Integral Formulation

To develop our algorithms, we will need to consider a more general development of the path integral approach to stochastic optimal control than presented in Kappen (2007) and Broek et al. (2008). In particular, we have to address that in many stochastic dynamical systems, the control transition matrix $\mathbf{G}_t$ is state dependent and its structure depends on the partition of the state in directly and

non-directly actuated parts. Since only some of the states are directly controlled, the state vector is partitioned into $\mathbf{x} = [\mathbf{x}^{(m)^T} \;\; \mathbf{x}^{(c)^T}]^T$ with $\mathbf{x}^{(m)} \in \Re^{k\times 1}$ the non-directly actuated part and $\mathbf{x}^{(c)} \in \Re^{l\times 1}$the directly actuated part. Subsequently, the passive dynamics term and the control transition matrix can be partitioned as $\mathbf{f}_t = [\mathbf{f}_t^{(m)^T} \;\; \mathbf{f}_t^{(c)^T}]^T$ with $\mathbf{f}_m \in \Re^{k\times 1}$, $\mathbf{f}_c \in \Re^{l\times 1}$ and $\mathbf{G}_t = [0_{k\times p} \;\; \mathbf{G}_t^{(c)^T}]^T$ with $\mathbf{G}_t^{(c)} \in \Re^{l\times p}$. The discretized state space representation of such systems is given as:

$$\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} + \mathbf{f}_{t_i} dt + \mathbf{G}_{t_i} \left(\mathbf{u}_{t_i} dt + \sqrt{dt}\varepsilon_{t_i} \right),$$

or, in partitioned vector form:

$$\begin{pmatrix} \mathbf{x}_{t_{i+1}}^{(m)} \\ \mathbf{x}_{t_{i+1}}^{(c)} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{t_i}^{(m)} \\ \mathbf{x}_{t_i}^{(c)} \end{pmatrix} + \begin{pmatrix} \mathbf{f}_{t_i}^{(m)} \\ \mathbf{f}_{t_i}^{(c)} \end{pmatrix} dt + \begin{pmatrix} 0_{k\times p} \\ \mathbf{G}_{t_i}^{(c)} \end{pmatrix} \left(\mathbf{u}_{t_i} dt + \sqrt{dt}\varepsilon_{t_i} \right). \tag{12}$$

Essentially the stochastic dynamics are partitioned into controlled equations in which the state $\mathbf{x}_{t_{i+1}}^{(c)}$ is directly actuated and the uncontrolled equations in which the state $\mathbf{x}_{t_{i+1}}^{(m)}$ is not directly actuated. Since stochasticity is only added in the directly actuated terms (c) of (12), we can develop $p(\tau_i|\mathbf{x}_i)$ as follows.

$$\begin{aligned} p(\tau_i|\mathbf{x}_{t_i}) &= p(\tau_{i+1}|\mathbf{x}_{t_i}) \\ &= p(\mathbf{x}_{t_N}, ..., \mathbf{x}_{t_{i+1}}|\mathbf{x}_{t_i}) \\ &= \Pi_{j=i}^{N-1} p\left(\mathbf{x}_{t_{j+1}}|\mathbf{x}_{t_j}\right), \end{aligned}$$

where we exploited the fact that the start state $\mathbf{x}_{t_i}$ of a trajectory is given and does not contribute to its probability. For systems where the control has lower dimensionality than the state (12), the transition probabilities $p\left(\mathbf{x}_{t_{j+1}}|\mathbf{x}_{t_j}\right)$ are factorized as follows:

$$\begin{aligned} p\left(\mathbf{x}_{t_{j+1}}|\mathbf{x}_{t_j}\right) &= p\left(\mathbf{x}_{t_{j+1}}^{(m)}|\mathbf{x}_{t_j}\right) \cdot p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}\right) \\ &= p\left(\mathbf{x}_{t_{j+1}}^{(m)}|\mathbf{x}_{t_j}^{(m)}, \mathbf{x}_{t_j}^{(c)}\right) \cdot p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}^{(m)}, \mathbf{x}_{t_j}^{(c)}\right) \\ &\propto p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}\right), \end{aligned} \tag{13}$$

where we have used the fact that $p\left(\mathbf{x}_{t_{i+1}}^{(m)}|\mathbf{x}_{t_i}^{(m)}, \mathbf{x}_{t_i}^{(c)}\right)$ is the Dirac delta function, since $\mathbf{x}_{t_{j+1}}^{(m)}$ can be computed deterministically from $\mathbf{x}_{t_j}^{(m)}, \mathbf{x}_{t_j}^{(c)}$. For all practical purposes,[2] the transition probability of the stochastic dynamics is reduced to the transition probability of the directly actuated part of the state:

$$p(\tau_i|\mathbf{x}_{t_i}) = \Pi_{j=i}^{N-1} p\left(\mathbf{x}_{t_{j+1}}|\mathbf{x}_{t_j}\right) \propto \Pi_{j=i}^{N-1} p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}\right). \tag{14}$$

Since we assume that the noise ε is zero mean Gaussian distributed with variance Σ_ε, where $\Sigma_\varepsilon \in \Re^{l\times l}$, the transition probability of the directly actuated part of the state is defined as:[3]

$$p\left(\mathbf{x}_{t_{j+1}}^{(c)}|\mathbf{x}_{t_j}\right) = \frac{1}{\left((2\pi)^l \cdot |\Sigma_{t_j}|\right)^{1/2}} \exp\left(-\frac{1}{2} \left\| \mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)} - \mathbf{f}_{t_j}^{(c)} dt \right\|_{\Sigma_{t_j}^{-1}}^2 \right), \tag{15}$$

2. The delta functions will all integrate to 1 in the path integral.

3. For notational simplicity, we write weighted square norms (or Mahalanobis distances) as $\mathbf{v}^T\mathbf{M}\mathbf{v} = \|\mathbf{v}\|_{\mathbf{M}}^2$.

where the covariance $\Sigma_{t_j} \in \Re^{l \times l}$ is expressed as $\Sigma_{t_j} = \mathbf{G}_{t_j}^{(c)} \Sigma_\varepsilon \mathbf{G}_{t_j}^{(c)^T} dt$. Combining (15) and (14) results in the probability of a path expressed as:

$$p(\tau_i|\mathbf{x}_{t_i}) \propto \frac{1}{\Pi_{j=i}^{N-1} \left((2\pi)^l |\Sigma_{t_j}|\right)^{1/2}} \exp\left(-\frac{1}{2}\sum_{j=1}^{N-1} \left\| \mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)} - \mathbf{f}_{t_j}^{(c)} dt \right\|_{\Sigma_{t_j}^{-1}}^2\right).$$

Finally, we incorporate the assumption (8) about the relation between the control cost and the variance of the noise, which needs to be adjusted to the controlled space as $\Sigma_{t_j} = \mathbf{G}_{t_j}^{(c)} \Sigma_\varepsilon \mathbf{G}_{t_j}^{(c)^T} dt = \lambda \mathbf{G}_{t_j}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_j}^{(c)^T} dt = \lambda \mathbf{H}_{t_j} dt$ with $\mathbf{H}_{t_j} = \mathbf{G}_{t_j}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_j}^{(c)^T}$. Thus, we obtain:

$$p(\tau_i|\mathbf{x}_{t_i}) \propto \frac{1}{\Pi_{j=i}^{N-1} \left((2\pi)^l |\Sigma_{t_j}|\right)^{1/2}} \exp\left(-\frac{1}{2\lambda}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|_{\mathbf{H}_{t_j}^{-1}}^2 dt\right).$$

With this formulation of the probability of a trajectory, we can rewrite the the path integral (11) as:

$$\begin{aligned}
\Psi_{t_i} &= \lim_{dt\to 0} \int \frac{\exp\left(-\frac{1}{\lambda}\left(\phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|_{\mathbf{H}_{t_j}^{-1}}^2 dt\right)\right)}{\Pi_{j=i}^{N-1} \left((2\pi)^{l/2} |\Sigma_{t_j}|^{1/2}\right)} d\tau_i^{(c)} \\
&= \lim_{dt\to 0} \int \frac{1}{D(\tau_i)} \exp\left(-\frac{1}{\lambda} S(\tau_i)\right) d\tau_i^{(c)}, \qquad (16)
\end{aligned}$$

where, we defined

$$S(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|_{\mathbf{H}_{t_j}^{-1}}^2 dt,$$

and

$$D(\tau_i) = \Pi_{j=i}^{N-1} \left((2\pi)^{l/2} |\Sigma_{t_j}|^{1/2}\right).$$

Note that the integration is over $d\tau_i^{(c)} = \left(d\mathbf{x}_{t_i}^{(c)}, ..., d\mathbf{x}_{t_N}^{(c)}\right)$, as the non-directly actuated states can be integrated out due to the fact that the state transition of the non-directly actuated states is deterministic, and just added Dirac delta functions in the integral (cf. Equation (13)). Equation (16) is written in a more compact form as:

$$\begin{aligned}
\Psi_{t_i} &= \lim_{dt\to 0} \int \exp\left(-\frac{1}{\lambda} S(\tau_i) - \log D(\tau_i)\right) d\tau_i^{(c)} \\
&= \lim_{dt\to 0} \int \exp\left(-\frac{1}{\lambda} Z(\tau_i)\right) d\tau_i^{(c)}, \qquad (17)
\end{aligned}$$

where $Z(\tau_i) = S(\tau_i) + \lambda \log D(\tau_i)$. It can be shown that this term is factorized in path dependent and path independent terms of the form:

$$Z(\tau_i) = \tilde{S}(\tau_i) + \frac{\lambda(N-i)l}{2}\log(2\pi dt\lambda),$$

where $\tilde{S}(\tau_i) = S(\tau_i) + \frac{\lambda}{2}\sum_{j=i}^{N-1}\log|\mathbf{H}_{t_j}|$. This formula is a required step for the derivation of optimal controls in the next section. The constant term $\frac{\lambda(N-i)l}{2}\log(2\pi dt\lambda)$ can be the source of numerical instabilities especially in cases where fine discretization dt of stochastic dynamics is required. However, in the next section, and in a great detail in Appendix A, lemma 1, we show how this term drops out of the equations.

2.4 Optimal Controls

For every moment of time, the optimal controls are given as $\mathbf{u}_{t_i} = -\mathbf{R}^{-1}\mathbf{G}_{t_i}^T(\nabla_{x_{t_i}}V_{t_i})$. Due to the exponential transformation of the value function, the equation of the optimal controls can be written as

$$\mathbf{u}_{t_i} = \lambda\mathbf{R}^{-1}\mathbf{G}_{t_i}\frac{\nabla_{\mathbf{x}_{t_i}}\Psi_{t_i}}{\Psi_{t_i}}.$$

After substituting Ψ_{t_i} with (17) and canceling the state independent terms of the cost we have:

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\left(\lambda\mathbf{R}^{-1}\mathbf{G}_{t_i}^T\frac{\nabla_{\mathbf{x}_{t_i}^{(c)}}\left(\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}d\tau_i^{(c)}\right)}{\int\ e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}d\tau_i^{(c)}}\right),$$

Further analysis of the equation above leads to a simplified version for the optimal controls as

$$\boxed{\mathbf{u}_{t_i} = \int P(\tau_i)\,\mathbf{u}_L(\tau_i)\,d\tau_i^{(c)}}, \tag{18}$$

with the probability $P(\tau_i)$ and local controls $\mathbf{u}_L(\tau_i)$ defined as

$$\boxed{P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}d\tau_i}} \tag{19}$$

$$\mathbf{u}_L(\tau_i) = -\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\lim_{dt\to 0}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right).$$

The path cost $\tilde{S}(\tau_i)$ is a generalized version of the path cost in Kappen (2005a) and Kappen (2007), which only considered systems with state independent control transition[4] $\mathbf{G}_{t_i}$. To find the local controls $\mathbf{u}_L(\tau_i)$ we have to calculate the $\lim_{dt\to 0}\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)$. Appendix A and more precisely lemma 2 shows in detail the derivation of the final result:

$$\lim_{dt\to 0}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right) = -\mathbf{H}_{t_i}^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right),$$

where the new term $\mathbf{b}_{t_i}$ is expressed as $\mathbf{b}_{t_i} = \lambda\mathbf{H}_{t_i}\Phi_{t_i}$ and $\Phi_{t_i} \in \Re^{l\times 1}$ is a vector with the j^{th} element defined as:

$$(\Phi_{t_i})_j = \frac{1}{2}trace\left(\mathbf{H}_{t_i}^{-1}\left(\partial_{[\mathbf{x}_{t_i}^{(c)}]_j}\mathbf{H}_{t_i}\right)\right).$$

4. More precisely if $\mathbf{G}_{t_i}^{(c)} = \mathbf{G}^{(c)}$ then the term $\frac{\lambda}{2}\sum_{j=i}^{N-1}\log|\mathbf{H}_{t_j}|$ disappears since it is state independent and it appears in both nominator and denominator in (19). In this case, the path cost is reduced to $\tilde{S}(\tau_i) = S(\tau_i)$.

The local control can now be expressed as:

$$\mathbf{u}_L(\tau_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right),$$

By substituting $\mathbf{H}_{t_i} = \mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}$ in the equation above, we get our main result for the local controls of the sampled path for the generalized path integral formulation:

$$\boxed{\mathbf{u}_L(\tau_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\left(\mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\right)^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right)}. \tag{20}$$

The equations in boxes (18), (19) and (20) form the solution for the generalized path integral stochastic optimal control problem. Given that this result is of general value and constitutes the foundation to derive our reinforcement learning algorithm in the next section, but also since many other special cases can be derived from it, we summarized all relevant equations in Table 1.

The **Given** components of Table 1 include a model of the system dynamics, the cost function, knowledge of the system's noise process, and a mechanism to generate trajectories τ_i. It is important to realize that this is a *model-based* approach, as the computations of the optimal controls requires knowledge of ε_i. ε_i can be obtained in two ways. First, the trajectories τ_i can be generated purely in simulation, where the noise is generated from a random number generator. Second, trajectories could be generated by a real system, and the noise ε_i would be computed from the difference between the actual and the predicted system behavior, that is, $\mathbf{G}_{t_i}^{(c)}\varepsilon_i = \dot{\mathbf{x}}_{t_i} - \hat{\dot{\mathbf{x}}}_{t_i} = \dot{\mathbf{x}}_{t_i} - (\mathbf{f}_{t_i} + \mathbf{G}_{t_i}\mathbf{u}_{t_i})$. Computing the prediction $\hat{\dot{\mathbf{x}}}_{t_i}$ also requires a model of the system dynamics.

Previous results in Kappen (2005a), Kappen (2007), Kappen (2005b) and Broek et al. (2008) are special cases of our generalized formulation. In the next section we show how our generalized formulation is specialized to different classes of stochastic dynamical systems and we provide the corresponding formula of local controls for each class.

2.5 Special Cases

The purpose of this section is twofold. First, it demonstrates how to apply the path integral approach to specialized forms of dynamical systems, and how the local controls in (20) simplify for these cases. Second, this section prepares the special case which we will need for our reinforcement learning algorithm in Section 3.

2.5.1 SYSTEMS WITH ONE DIMENSIONAL DIRECTLY ACTUATED STATE

The generalized formulation of stochastic optimal control with path integrals in Table 1 can be applied to a variety of stochastic dynamical systems with different types of control transition matrices. One case of particular interest is where the dimensionality of the directly actuated part of the state is 1D, while the dimensionality of the control vector is 1D or higher dimensional. As will be seen below, this situation arises when the controls are generated by a linearly parameterized function approximator. The control transition matrix thus becomes a row vector $\mathbf{G}_{t_i}^{(c)} = \mathbf{g}_{t_i}^{(c)T} \in \Re^{1\times p}$. According to (20), the local controls for such systems are expressed as follows:

$$\mathbf{u}_L(\tau_i) = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)}}{\mathbf{g}_{t_i}^{(c)T}\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)}}\left(\mathbf{g}_{t_i}^{(c)T}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right).$$

- **Given**:
  - The system dynamics $\dot{\mathbf{x}}_t = \mathbf{f}_t + \mathbf{G}_t\left(\mathbf{u}_t + \varepsilon_t\right)$ (cf. 3)
  - The immediate cost $r_t = q_t + \frac{1}{2}\mathbf{u}_t^T \mathbf{R}\mathbf{u}_t$ (cf. 4)
  - A terminal cost term ϕ_{t_N} (cf. 1)
  - The variance Σ_ε of the mean-zero noise ε_t
  - Trajectory starting at t_i and ending at t_N: $\tau_i = (\mathbf{x}_{t_i}, ..., \mathbf{x}_{t_N})$
  - A partitioning of the system dynamics into (c) controlled and (m) uncontrolled equations, where $n = c + m$ is the dimensionality of the state $\mathbf{x}_t$ (cf. Section 2.3)
- **Optimal Controls**:
  - Optimal controls at every time step t_i: $\mathbf{u}_{t_i} = \int P\left(\tau_i\right)\mathbf{u}\left(\tau_i\right) d\tau_i^{(c)}$
  - Probability of a trajectory: $P\left(\tau_i\right) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i}$
  - Generalized trajectory cost: $\tilde{S}(\tau_i) = S(\tau_i) + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log|\mathbf{H}_{t_j}|$ where
    - $S(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|_{\mathbf{H}_{t_j}^{-1}}^2 dt$
    - $\mathbf{H}_{t_j} = \mathbf{G}_{t_j}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_j}^{(c)T}$
  - Local Controls: $\mathbf{u}_L(\tau_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\left(\mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\right)^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right)$ where
    - $\mathbf{b}_{t_i} = \lambda \mathbf{H}_{t_i}\Phi_{t_i}$
    - $[\Phi_{t_i}]_j = \frac{1}{2} trace\left(\mathbf{H}_{t_i}^{-1}\left(\partial_{[\mathbf{x}_{t_i}^{(c)}]_j}\mathbf{H}_{t_i}\right)\right)$

Table 1: Summary of optimal control derived from the path integral formalizm.

Since the directly actuated part of the state is 1D, the vector $\mathbf{x}_{t_i}^{(c)}$ collapses into the scalar $x_{t_i}^{(c)}$ which appears in the partial differentiation above. In the case that $\mathbf{g}_{t_i}^{(c)}$ does not depend on $x_{t_i}^{(c)}$, the differentiation with respect to $x_{t_i}^{(c)}$ results to zero and the the local controls simplify to:

$$\mathbf{u}_L(\tau_i) = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)}\mathbf{g}_{t_i}^{(c)T}}{\mathbf{g}_{t_i}^{(c)T}\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)}}\varepsilon_{t_i}.$$

2.5.2 SYSTEMS WITH PARTIALLY ACTUATED STATE

The generalized formula of the local controls (20) was derived for the case where the control transition matrix is state dependent and its dimensionality is $\mathbf{G}_t^{(c)} \in \Re^{l \times p}$ with $l < n$ and p the dimensionality of the control. There are many special cases of stochastic dynamical systems in optimal control

and robotic applications that belong into this general class. More precisely, for systems having a state dependent control transition matrix that is square ($\mathbf{G}_{t_i}^{(c)} \in \Re^{l \times l}$ with $l = p$) the local controls based on (20) are reformulated as:

$$\mathbf{u}_L(\tau_i) = \varepsilon_{t_i} - \mathbf{G}_{t_i}^{(c)^{-1}} \mathbf{b}_{t_i}. \tag{21}$$

Interestingly, a rather general class of mechanical systems such as rigid-body and multi-body dynamics falls into this category. When these mechanical systems are expressed in state space formulation, the control transition matrix is equal to rigid body inertia matrix $\mathbf{G}_{t_i}^{(c)} = \mathbf{M}(\theta_{t_i})$ (Sciavicco and Siciliano, 2000). Future work will address this special topic of path integral control for multi-body dynamics.

Another special case of systems with partially actuated state is when the control transition matrix is state independent and has dimensionality $\mathbf{G}_t^{(c)} = \mathbf{G}^{(c)} \in \Re^{l \times p}$. The local controls, according to (20), become:

$$\mathbf{u}_L(\tau_i) = \mathbf{R}^{-1}\mathbf{G}^{(c)^T}\left(\mathbf{G}^{(c)}\mathbf{R}^{-1}\mathbf{G}^{(c)^T}\right)^{-1}\mathbf{G}^{(c)}\varepsilon_{t_i}. \tag{22}$$

If $\mathbf{G}_{t_i}^{(c)}$ is square and state independent, $\mathbf{G}_{t_i}^{(c)} = \mathbf{G}^{(c)} \in \Re^{l \times l}$, we will have:

$$\mathbf{u}_L(\tau_i) = \varepsilon_{t_i}. \tag{23}$$

This special case was explored in Kappen (2005a), Kappen (2007), Kappen (2005b) and Broek et al. (2008). Our generalized formulation allows a broader application of path integral control in areas like robotics and other control systems, where the control transition matrix is typically partitioned into directly and non-directly actuated states, and typically also state dependent.

2.5.3 SYSTEMS WITH FULLY ACTUATED STATE SPACE

In this class of stochastic systems, the control transition matrix is not partitioned and, therefore, the control $\mathbf{u}$ directly affects all the states. The local controls for such systems are provided by simply substituting $\mathbf{G}_{t_i}^{(c)} \in \Re^{n \times p}$ in (20) with $\mathbf{G}_{t_i} \in \Re^{n \times n}$. Since $\mathbf{G}_{t_i}$ is a square matrix we obtain:

$$\mathbf{u}_L(\tau_i) = \varepsilon_{t_i} - \mathbf{G}_{t_i}^{-1}\mathbf{b}_{t_i},$$

with $\mathbf{b}_{t_i} = \lambda \mathbf{H}_{t_i}\Phi_{t_i}$ and

$$(\Phi_{t_i})_j = \frac{1}{2} trace\left(\mathbf{H}_{t_i}^{-1}\left(\partial_{(\mathbf{x}_{t_i})_j}\mathbf{H}_{t_i}\right)\right),$$

where the differentiation is not taken with respect to $(\mathbf{x}_{t_i}^{(c)})_j$ but with respect to the full state $(\mathbf{x}_{t_i})_j$. For this fully actuated state space, there are subclasses of dynamical systems with square and/or state independent control transition matrix. The local controls for these cases are found by just substituting $\mathbf{G}_{t_i}^{(c)}$ with $\mathbf{G}_{t_i}$ in (21), (22) and (23).

3. Reinforcement Learning with Parameterized Policies

Equipped with the theoretical framework of stochastic optimal control with path integrals, we can now turn to its application to reinforcement learning with parameterized policies. Since the beginning of actor-critic algorithms (Barto et al., 1983), one goal of reinforcement learning has been

to learn compact policy representations, for example, with neural networks as in the early days of machine learning (Miller et al., 1990), or with general parameterizations (Peters, 2007; Deisenroth et al., 2009). Parameterized policies have much fewer parameters than the classical time-indexed approach of optimal control, where every time step has it own set of parameters, that is, the optimal controls at this time step. Usually, function approximation techniques are used to represent the optimal controls and the open parameters of the function approximator become the policy parameters. Function approximators use a state representation as input and not an explicit time dependent representation. This representation allows generalization across states and promises to achieve better generalization of the control policy to a larger state space, such that policies become re-usable and do not have to be recomputed in every new situation.

The path integral approach from the previous sections also follows the classical time-based optimal control strategy, as can be seen from the time dependent solution for optimal controls in (33). However, a minor re-interpretation of the approach and some small mathematical adjustments allow us to carry it over to parameterized policies and reinforcement learning, which results in a new algorithm called **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}).

3.1 Parameterized Policies

We are focusing on direct policy learning, where the parameters of the policy are adjusted by a learning rule directly, and not indirectly as in value function approaches of classical reinforcement learning (Sutton and Barto, 1998)—see Peters (2007) for a discussion of pros and cons of direct vs. indirect policy learning. Direct policy learning usually assumes a general cost function (Sutton et al., 2000; Peters, 2007) in the form of

$$J(\mathbf{x}_0) = \int p(\tau_0) R(\tau_0) d\tau_0, \tag{24}$$

which is optimized over states-action trajectories[5] $\tau_0 = (\mathbf{x}_{t_0}, \mathbf{a}_{t_0}, ..., \mathbf{x}_{t_N})$. Under the first order Markov property, the probability of a trajectory is

$$p(\tau_i) = p(\mathbf{x}_{t_i}) \Pi_{j=i}^{N-1} p(\mathbf{x}_{t_{j+1}} | \mathbf{x}_{t_j}, \mathbf{a}_{t_j}) p(\mathbf{a}_{t_j} | \mathbf{x}_{t_j}).$$

Both the state transition and the policy are assumed to be stochastic. The particular formulation of the stochastic policy is a design parameter, motivated by the application domain, analytical convenience, and the need to inject exploration during learning. For continuous state action domains, Gaussian distributions are most commonly chosen (Gullapalli, 1990; Williams, 1992; Peters, 2007). An interesting generalized stochastic policy was suggested in Rueckstiess et al. (2008) and applied in Koeber and Peters (2008), where the stochastic policy $p(\mathbf{a}_{t_i} | \mathbf{x}_{t_i})$ is linearly parameterized as:

$$\mathbf{a}_{t_i} = \mathbf{g}_{t_i}^T (\theta + \varepsilon_{t_i}), \tag{25}$$

with $\mathbf{g}_{t_i}$ denoting a vector of basis functions and θ the parameter vector. This policy has state dependent noise, which can contribute to faster learning as the signal-to-noise ratio becomes adaptive since it is a function of $\mathbf{g}_{t_i}$. It should be noted that a standard additive-noise policy can be expressed in this formulation, too, by choosing one basis function $(\mathbf{g}_{t_i})_j = 0$. For Gaussian noise ε the probability of an action is $p(\mathbf{a}_{t_i} | \mathbf{x}_{t_i}) = N\left(\theta^T \mathbf{g}_{t_i}, \Sigma_{t_i}\right)$ with $\Sigma_{t_i} = \mathbf{g}_{t_i}^T \Sigma_\varepsilon \mathbf{g}_{t_i}$. Comparing the policy formulation

5. We use $\mathbf{a}_t$ to denote actions here in order to avoid using the symbol $\mathbf{u}$ in a conflicting way in the equations below, and to emphasize that an action does not necessarily coincide with the control command to a physical system.

in (25) with the control term in (3), one recognizes that the control policy formulation (25) should fit into the framework of path integral optimal control.

3.2 Generalized Parameterized Policies

Before going into more detail of our proposed reinforcement learning algorithm, it is worthwhile contemplating what the action $\mathbf{a}_t$ actually represents. In many applications of stochastic optimal control there are three main problems to be considered: i) trajectory planning, ii) feedforward control, and iii) feedback control. The results of optimization could thus be an optimal kinematic trajectory, the corresponding feedforward commands to track the desired trajectory accurately in face of the system's nonlinearities, and/or time varying linear feedback gains (gain scheduling) for a negative feedback controller that compensates for perturbations from accurate trajectory tracking.

There are very few optimal control algorithms which compute all three issues simultaneously, such as Differential Dynamic Programming(DDP) (Jacobson and Mayne, 1970), or its simpler version the Iterative Linear Quadratic Regulator(iLQR) (Todorov, 2005). However, these are model based methods which require rather accurate knowledge of the dynamics and make restrictive assumptions concerning differentiability of the system dynamics and the cost function.

Path integral optimal control allows more flexibility than these related methods. The concept of an "action" can be viewed in a broader sense. Essentially, we consider any "input" to the control system as an action, not unlike the inputs to a transfer function in classical linear control theory. The input can be a motor command, but it can also be anything else, for instance, a desired state, that is subsequently converted to a motor command by some tracking controller, or a control gain (Buchli et al., 2010) . As an example, consider a robotic system with rigid body dynamics (RBD) equations (Sciavicco and Siciliano, 2000) using a parameterized policy:

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1}\left(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})-\mathbf{v}(\mathbf{q})\right)+\mathbf{M}(\mathbf{q})^{-1}\mathbf{u}, \tag{26}$$

$$\mathbf{u} = \mathbf{G}(\mathbf{q})(\theta+\varepsilon_{t_i}), \tag{27}$$

where $\mathbf{M}$ is the RBD inertia matrix, $\mathbf{C}$ are Coriolis and centripetal forces, and $\mathbf{v}$ denotes gravity forces. The state of the robot is described by the joint angles $\mathbf{q}$ and joint velocities $\dot{\mathbf{q}}$. The policy (27) is linearly parameterized by θ, with basis function matrix $\mathbf{G}$—one would assume that the dimensionality of θ is significantly larger than that of $\mathbf{q}$ to assure sufficient expressive power of this parameterized policy. Inserting (27) into (26) results in a differential equation that is compatible with the system equations (3) for path integral optimal control:

$$\begin{aligned} \ddot{\mathbf{q}} &= \mathbf{f}(\mathbf{q},\dot{\mathbf{q}})+\tilde{\mathbf{G}}(\mathbf{q})(\theta+\varepsilon_{t_i}) \\ &\textit{where} \\ \mathbf{f}(\mathbf{q},\dot{\mathbf{q}}) &= \mathbf{M}(\mathbf{q})^{-1}\left(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})-\mathbf{v}(\mathbf{q})\right), \\ \tilde{\mathbf{G}}(\mathbf{q}) &= \mathbf{M}(\mathbf{q})^{-1}\mathbf{G}(\mathbf{q}). \end{aligned} \tag{28}$$

This example is a typical example where the policy directly represents motor commands.

Alternatively, we could create another form of control structure for the RBD system:

$$\begin{aligned} \ddot{\mathbf{q}} &= \mathbf{M}(\mathbf{q})^{-1}\left(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})-\mathbf{v}(\mathbf{q})\right)+\mathbf{M}(\mathbf{q})^{-1}\mathbf{u}, \\ \mathbf{u} &= \mathbf{K}_P(\mathbf{q}_d-\mathbf{q})+\mathbf{K}_D(\dot{\mathbf{q}}_d-\dot{\mathbf{q}}), \\ \ddot{\mathbf{q}}_d &= \mathbf{G}(\mathbf{q}_d,\dot{\mathbf{q}}_d)(\theta+\varepsilon_{t_i}). \end{aligned} \tag{29}$$

Here, a Proportional-Derivative (PD) controller with positive definite gain matrices $\mathbf{K}_P$ and $\mathbf{K}_D$ converts a desired trajectory $\mathbf{q}_d, \dot{\mathbf{q}}_d$ into a motor command $\mathbf{u}$. In contrast to the previous example, the parameterized policy generates the desired trajectory in (29), and the differential equation for the desired trajectory is compatible with the path integral formalism.

What we would like to emphasize is that the control system's structure is left to the creativity of its designer, and that path integral optimal control can be applied on various levels. Importantly, as developed in Section 2.3, only the *controlled* differential equations of the entire control system contribute to the path integral formalism, that is, (28) in the first example, or (29) in the second example. And *only these controlled differential equations* need to be known for applying path integral optimal control—none of the variables of the uncontrolled equations is ever used.

At this point, we make a very important transition from model-based to model-free learning. In the example of (28), the dynamics model of the control system needs to be known to apply path integral optimal control, as this is a *controlled* differential equation. In contrast, in (29), the system dynamics are in an *uncontrolled* differential equation, and are thus irrelevant for applying path integral optimal control. In this case, only knowledge of the desired trajectory dynamics is needed, which is usually created by the system designer. Thus, we obtained a model-free learning system.

3.3 Dynamic Movement Primitives as Generalized Policies

As we are interested in model-free learning, we follow the control structure of the 2^{nd} example of the previous section, that is, we optimize control policies which represent desired trajectories. We use Dynamic Movement Primitives (DMPs) (Ijspeert et al., 2003) as a special case of parameterized policies, which are expressed by the differential equations:

$$\begin{aligned} \frac{1}{\tau}\dot{z}_t &= f_t + \mathbf{g}_t^T(\theta+\varepsilon_t), \\ \frac{1}{\tau}\dot{y}_t &= z_t, \\ \frac{1}{\tau}\dot{x}_t &= -\alpha x_t, \\ f_t &= \alpha_z(\beta_z(g-y_t)-z_t). \end{aligned} \tag{30}$$

Essentially, these policies code a learnable point attractor for a movement from y_{t_0} to the goal g, where θ determines the shape of the attractor. $y_t, \dot{y}_t$ denote the position and velocity of the trajectory, while z_t, x_t are internal states. α_z, β_z, τ are time constants. The basis functions $\mathbf{g}_t \in \Re^{p\times 1}$are defined by a piecewise linear function approximator with Gaussian weighting kernels, as suggested in Schaal and Atkeson (1998):

$$\begin{aligned} [\mathbf{g}_t]_j &= \frac{w_j\, x_t}{\sum_{k=1}^{p} w_k}(g-y_0), \\ w_j &= \exp\left(-0.5h_j(x_t-c_j)^2\right), \end{aligned} \tag{31}$$

with bandwith h_j and center c_j of the Gaussian kernels—for more details see Ijspeert et al. (2003). The DMP representation is advantageous as it guarantees attractor properties towards the goal while remaining linear in the parameters θ of the function approximator. By varying the parameter θ the shape of the trajectory changes while the goal state g and initial state y_{t_0} remain fixed. These properties facilitate learning (Peters and Schaal, 2008a).

3.4 Policy Improvements with Path Integrals: The (PI^2) Algorithm

As can be easily recognized, the DMP equations are of the form of our control system (3), with only one controlled equation and a one dimensional actuated state. This case has been treated in Section 2.5.1. The motor commands are replaced with the parameters θ—the issue of time dependent vs. constant parameters will be addressed below. More precisely, the DMP equations can be written as:

$$\begin{pmatrix} \dot{x}_t \\ \dot{z}_t \\ \dot{y}_t \end{pmatrix} = \begin{pmatrix} -\alpha x_t \\ y_t \\ \alpha_z(\beta_z(g-y_t)-z_t) \end{pmatrix} + \begin{pmatrix} 0_{1\times p} \\ 0_{1\times p} \\ \mathbf{g}_t^{(c)T} \end{pmatrix} (\theta_t + \varepsilon_t).$$

The state of the DMP is partitioned into the controlled part $\mathbf{x}_t^{(c)} = y_t$ and uncontrolled part $\mathbf{x}_t^{(m)} = (x_t \; z_t)^T$. The control transition matrix depends on the state, however, it depends only on one of the state variables of the uncontrolled part of the state, that is, x_t. The path cost for the stochastic dynamics of the DMPs is given by:

$$\begin{aligned}
\tilde{S}(\tau_i) &= \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \right\|^2_{\mathbf{H}_{t_j}^{-1}} dt + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log|\mathbf{H}_{t_j}| \\
&\propto \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \left\| \mathbf{g}_{t_j}^{(c)T}(\theta_{t_j} + \varepsilon_{t_j}) \right\|^2_{\mathbf{H}_{t_j}^{-1}} \\
&= \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \frac{1}{2}(\theta_{t_j} + \varepsilon_{t_j})^T \mathbf{g}_{t_j}^{(c)} \mathbf{H}_{t_j}^{-1} \mathbf{g}_{t_j}^{(c)T}(\theta_{t_j} + \varepsilon_{t_j}) \\
&= \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \frac{1}{2}(\theta_{t_j} + \varepsilon_{t_j})^T \frac{\mathbf{g}_{t_j}^{(c)} \mathbf{g}_{t_j}^{(c)T}}{\mathbf{g}_t^{(c)T} \mathbf{R}^{-1} \mathbf{g}_t^{(c)}}(\theta_{t_j} + \varepsilon_{t_j}) \\
&= \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \frac{1}{2}(\theta_{t_j} + \varepsilon_{t_j})^T \mathbf{M}_{t_j}^T \mathbf{R} \mathbf{M}_{t_j}(\theta_{t_j} + \varepsilon_{t_j}).
\end{aligned} \tag{32}$$

with $\mathbf{M}_{t_j} = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_j}\mathbf{g}_{t_j}^T}{\mathbf{g}_{t_j}^T \mathbf{R}^{-1} \mathbf{g}_{t_j}}$. $\mathbf{H}_t$ becomes a scalar given by $\mathbf{H}_t = \mathbf{g}_t^{(c)T} \mathbf{R}^{-1} \mathbf{g}_t^{(c)}$. Interestingly, the term $\frac{\lambda}{2}\sum_{j=i}^{N-1} \log|\mathbf{H}_{t_j}|$ for the case of DMPs depends only on x_t, which is a deterministic variable and therefore can be ignored since it is the same for all sampled paths. We also absorbed, without loss of generality, the time step dt in cost terms. Consequently, the fundamental result of the path integral stochastic optimal problem for the case of DMPs is expressed as:

$$\boxed{\mathbf{u}_{t_i} = \int P(\tau_i)\, \mathbf{u}_L(\tau_i)\, d\tau_i^{(c)}}, \tag{33}$$

where the probability $P(\tau_i)$ and local controls $\mathbf{u}(\tau_i)$ are defined as

$$P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i}, \quad \mathbf{u}_L(\tau_i) = \frac{\mathbf{R}^{-1}\mathbf{g}_{t_i}^{(c)} \mathbf{g}_{t_i}^{(c)T}}{\mathbf{g}_{t_i}^{(c)T} \mathbf{R}^{-1} \mathbf{g}_{t_i}^{(c)}} \varepsilon_{t_i},$$

and the path cost given as

$$\tilde{S}(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2} \sum_{j=i}^{N-1} \varepsilon_{t_j}^T \mathbf{M}_{t_j}^T \mathbf{R} \mathbf{M}_{t_j} \varepsilon_{t_j}.$$

Note that $\theta = 0$ in these equations, that is, the parameters are initialized to zero. These equations correspond to the case where the stochastic optimal control problem is solved with one evaluation of the optimal controls (33) using dense sampling of the whole state space under the "passive dynamics" (i.e., $\theta = 0$), which requires a significant amount of exploration noise. Such an approach was pursued in the original work by Kappen (2007) and Broek et al. (2008), where a potentially large number of sample trajectories was needed to achieve good results. Extending this sampling approach to high dimensional spaces, however, is daunting, as with very high probability, we would sample primarily rather useless trajectories. Thus, biasing sampling towards good initial conditions seems to be mandatory for high dimensional applications.

Thus, we consider only local sampling and an iterative update procedure. Given a current guess of θ, we generate sample roll-outs using stochastic parameters $\theta + \varepsilon_t$ at every time step. To see how the generalized path integral formulation is modified for the case of iterative updating, we start with the equations of the update of the parameter vector θ, which can be written as:

$$\begin{aligned}
\theta_{t_i}^{(new)} &= \int P(\tau_i) \frac{\mathbf{R}^{-1} \mathbf{g}_{t_i} \mathbf{g}_{t_i}^T (\theta + \varepsilon_{t_i})}{\mathbf{g}_{t_i}^T \mathbf{R}^{-1} \mathbf{g}_{t_i}} d\tau_i \\
&= \int P(\tau_i) \frac{\mathbf{R}^{-1} \mathbf{g}_{t_i} \mathbf{g}_{t_i}^T \varepsilon_{t_i}}{\mathbf{g}_{t_i}^T \mathbf{R}^{-1} \mathbf{g}_{t_i}} d\tau_i + \frac{\mathbf{R}^{-1} \mathbf{g}_{t_i} \mathbf{g}_{t_i}^T \theta}{\mathbf{g}_{t_i}^T \mathbf{R}^{-1} \mathbf{g}_{t_i}} \\
&= \delta\theta_{t_i} + \frac{\mathbf{R}^{-1} \mathbf{g}_{t_i} \mathbf{g}_{t_i}^T}{trace(\mathbf{R}^{-1} \mathbf{g}_{t_i} \mathbf{g}_{t_i}^T)} \theta \\
&= \delta\theta_{t_i} + \mathbf{M}_{t_i} \theta. \qquad (34)
\end{aligned}$$

The correction parameter vector $\delta\theta_{t_i}$ is defined as $\delta\theta_{t_i} = \int P(\tau_i) \frac{\mathbf{R}^{-1} \mathbf{g}_{t_i} \mathbf{g}_{t_i}^T \varepsilon_{t_i}}{\mathbf{g}_{t_i}^T \mathbf{R}^{-1} \mathbf{g}_{t_i}} d\tau_i$. It is important to note that $\theta_{t_i}^{(new)}$ is now time dependent, that is, for every time step t_i, a different optimal parameter vector is computed. In order to return to one single time independent parameter vector $\theta^{(new)}$, the vectors $\theta_{t_i}^{(new)}$ need to be averaged over time t_i.

We start with a first tentative suggestion of averaging over time, and then explain why it is inappropriate, and what the correct way of time averaging has to look like. The tentative and most intuitive time average is:

$$\theta^{(new)} = \frac{1}{N} \sum_{i=0}^{N-1} \theta_{t_i}^{(new)} = \frac{1}{N} \sum_{i=0}^{N-1} \delta\theta_{t_i} + \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{M}_{t_i} \theta.$$

Thus, we would update θ based on two terms. The first term is the average of $\delta\theta_{t_i}$, which is reasonable as it reflects the knowledge we gained from the exploration noise. However, there would be a second update term due to the average over projected mean parameters θ from every time step—it should be noted that $\mathbf{M}_{t_i}$ is a projection matrix onto the range space of $\mathbf{g}_{t_i}$ under the metric $\mathbf{R}^{-1}$, such that a multiplication with $\mathbf{M}_{t_i}$ can only shrink the norm of θ. From the viewpoint of having optimal parameters for *every time step*, this update component is reasonable as it trivially eliminates the part of the parameter vector that lies in the null space of $\mathbf{g}_{t_i}$ and which contributes to the command cost

of a trajectory in a useless way. From the view point of a parameter vector that is *constant and time independent* and that is updated *iteratively*, this second update is undesirable, as the multiplication of the parameter vector θ with $\mathbf{M}_{t_i}$ in (34) and the averaging operation over the time horizon reduces the L_2 norm of the parameters at every iteration, potentially in an uncontrolled way.[6] What we rather want is to achieve convergence when the average of $\delta\theta_{t_i}$ becomes zero, and we do not want to continue updating due to the second term.

The problem is avoided by eliminating the projection matrix in the second term of averaging, such that it become:

$$\theta^{(new)} = \frac{1}{N}\sum_{i=0}^{N-1}\delta\theta_{t_i} + \frac{1}{N}\sum_{i=0}^{N-1}\theta = \frac{1}{N}\sum_{i=0}^{N-1}\delta\theta_{t_i} + \theta.$$

The meaning of this reduced update is simply that we keep a component in θ that is irrelevant and contributes to our trajectory cost in a useless way. However, this irrelevant component will not prevent us from reaching the optimal effective solution, that is, the solution that lies in the range space of $\mathbf{g}_{t_i}$. Given this modified update, it is, however, also necessary to derive a compatible cost function. As mentioned before, in the unmodified scenario, the last term of (32) is:

$$\frac{1}{2}\sum_{j=i}^{N-1}(\theta+\varepsilon_{t_j})^T\mathbf{M}_{t_j}^T\mathbf{R}\mathbf{M}_{t_j}(\theta+\varepsilon_{t_j})$$

To avoid a projection of θ, we modify this cost term to be:

$$\frac{1}{2}\sum_{j=i}^{N-1}(\theta+\mathbf{M}_{t_j}\varepsilon_{t_j})^T\mathbf{R}(\theta+\mathbf{M}_{t_j}\varepsilon_{t_j}).$$

With this modified cost term, the path integral formalism results in the desired $\theta_{t_i}^{(new)}$ without the $\mathbf{M}_{t_i}$ projection of θ.

The main equations of the iterative version of the generalized path integral formulation, called **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}), can be summarized as:

$$P(\tau_i) = \frac{e^{-\frac{1}{\lambda}S(\tau_i)}}{\int e^{-\frac{1}{\lambda}S(\tau_i)}d\tau_i}, \tag{35}$$

$$S(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j}dt + \frac{1}{2}\sum_{j=i}^{N-1}(\theta+\mathbf{M}_{t_j}\varepsilon_{t_j})^T\mathbf{R}(\theta+\mathbf{M}_{t_j}\varepsilon_{t_j})dt, \tag{36}$$

$$\delta\theta_{t_i} = \int P(\tau_i)\,\mathbf{M}_{t_i}\varepsilon_{t_i}d\tau_i, \tag{37}$$

$$[\delta\theta]_j = \frac{\sum_{i=0}^{N-1}(N-i)\;w_{j,t_i}\;[\delta\theta_{t_i}]_j}{\sum_{i=0}^{N-1}w_{j,t_i}(N-i)}, \tag{38}$$

$$\theta^{(new)} = \theta^{(old)} + \delta\theta.$$

Essentially, (35) computes a discrete probability at time t_i of each trajectory roll-out with the help of the cost (36). For every time step of the trajectory, a parameter update is computed in (37) based

6. To be precise, θ would be projected and continue shrinking until it lies in the intersection of all null spaces of the $\mathbf{g}_{t_i}$ basis function—this null space can easily be of measure zero.

on a probability weighted average over trajectories. The parameter updates at every time step are finally averaged in (38). Note that we chose a weighted average by giving every parameter update a weight[7] according to the time steps left in the trajectory and the activation of the kernel in (31). This average can be interpreted as using a function approximator with only a constant (offset) parameter vector to approximate the time dependent parameters. Giving early points in the trajectory a higher weight is useful since their parameters affect a large time horizon and thus higher trajectory costs. Other function approximation (or averaging) schemes could be used to arrive at a final parameter update—we preferred this simple approach as it gave very good learning results. The final parameter update is $\theta^{(new)} = \theta^{(old)} + \delta\theta$.

The parameter λ regulates the sensitivity of the exponentiated cost and can automatically be optimized for every time step i to maximally discriminate between the experienced trajectories. More precisely, a constant term can be subtracted from (36) as long as all $S(\tau_i)$ remain positive—this constant term[8] cancels in (35). Thus, for a given number of roll-outs, we compute the exponential term in (35) as

$$\exp\left(-\frac{1}{\lambda}S(\tau_i)\right) = \exp\left(-h\frac{S(\tau_i)-\min S(\tau_i)}{\max S(\tau_i)-\min S(\tau_i)}\right),$$

with h set to a constant, which we chose to be $h = 10$ in all our evaluations. The max and min operators are over all sample roll-outs. This procedure eliminates λ and leaves the variance of the exploration noise ε as the only open algorithmic parameter for **PI**^{2}. It should be noted that the equations for **PI**^{2} have no numerical pitfalls: no matrix inversions and no learning rates,[9] rendering **PI**^{2} to be very easy to use in practice.

The pseudocode for the final **PI**^{2} algorithm for a one dimensional control system with function approximation is given in Table 2. A tutorial Matlab example of applying **PI**^{2} can be found at http://www-clmc.usc.edu/software .

4. Related Work

In the next sections we discuss related work in the areas of stochastic optimal control and reinforcement learning and analyze the connections and differences with the **PI**^{2} algorithm and the generalized path integral control formulation.

4.1 Stochastic Optimal Control and Path Integrals

The path integral formalism for optimal control was introduced in Kappen (2005a,b). In this work, the role of noise in symmetry breaking phenomena was investigated in the context of stochastic optimal control. In Kappen et al. (2007), Wiegerinck et al. (2006), and Broek et al. (2008), the path integral formalism is extended for the stochastic optimal control of multi-agent systems.

Recent work on stochastic optimal control by Todorov (2008), Todorov (2007) and Todorov (2009b) shows that for a class of discrete stochastic optimal control problems, the Bellman equa-

7. The use of the kernel weights in the basis functions (31) for the purpose of time averaging has shown better performance with respect to other weighting approaches, across all of our experiments. Therefore this is the weighting that we suggest. Users may develop other weighting schemes as more suitable to their needs.

8. In fact, the term inside the exponent results by adding $\frac{h\min S(\tau_i)}{\max S(\tau_i)-\min S(\tau_i)}$, which cancels in (35), to the term $-\frac{hS(\tau_i)}{\max S(\tau_i)-\min S(\tau_i)}$ which is equal to $-\frac{1}{\lambda}S(\tau_i)$.

9. **R** is a user design parameter and usually chosen to be diagonal and invertible.

- **Given**:
  - An immediate cost function $r_t = q_t + \theta_t^T \mathbf{R} \theta_t$ (cf. 1)
  - A terminal cost term ϕ_{t_N} (cf. 1)
  - A stochastic parameterized policy $\mathbf{a}_t = \mathbf{g}_t^T (\theta + \varepsilon_t)$ (cf. 25)
  - The basis function $\mathbf{g}_{t_i}$ from the system dynamics (cf. 3 and Section 2.5.1)
  - The variance Σ_ε of the mean-zero noise ε_t
  - The initial parameter vector θ
- **Repeat** until convergence of the trajectory cost R:
  - Create K roll-outs of the system from the same start state $\mathbf{x}_0$ using stochstic parameters $\theta + \varepsilon_t$ at every time step
  - **For** $k = 1...K$, compute:
    - $P(\tau_{i,k}) = \frac{e^{-\frac{1}{\lambda} S(\tau_{i,k})}}{\sum_{k=1}^{K} [e^{-\frac{1}{\lambda} S(\tau_{i,k})}]}$
    - $S(\tau_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j,k} + \frac{1}{2} \sum_{j=i+1}^{N-1} (\theta + \mathbf{M}_{t_j,k} \varepsilon_{t_j,k})^T \mathbf{R} (\theta + \mathbf{M}_{t_j,k} \varepsilon_{t_j,k})$
    - $\mathbf{M}_{t_j,k} = \frac{\mathbf{R}^{-1} \mathbf{g}_{t_j,k} \mathbf{g}_{t_j,k}^T}{\mathbf{g}_{t_j,k}^T \mathbf{R}^{-1} \mathbf{g}_{t_j,k}}$
  - **For** $i = 1...(N-1)$, compute:
    - $\delta\theta_{t_i} = \sum_{k=1}^{K} [P(\tau_{i,k}) \mathbf{M}_{t_i,k} \varepsilon_{t_i,k}]$
  - Compute $[\delta\theta]_j = \frac{\sum_{i=0}^{N-1} (N-i) w_{j,t_i} [\delta\theta_{t_i}]_j}{\sum_{i=0}^{N-1} w_{j,t_i} (N-i)}$
  - Update $\theta \leftarrow \theta + \delta\theta$
  - Create one noiseless roll-out to check the trajectory cost $R = \phi_{t_N} + \sum_{i=0}^{N-1} r_{t_i}$. In case the noise cannot be turned off, that is, a stochastic system, multiple roll-outs need be averaged.

Table 2: Pseudocode of the **PI**^{2} algorithm for a 1D Parameterized Policy (Note that the discrete time step dt was absorbed as a constant multiplier in the cost terms).

tion can be written as the KL divergence between the probability distribution of the controlled and uncontrolled dynamics. Furthermore it is shown that the class of discrete KL divergence control problem is equivalent to the continuous stochastic optimal control formalism with quadratic cost control function and under the presence of Gaussian noise. In Kappen et al. (2009), the KL divergence control formalism is considered and it is transformed to a probabilistic inference problem. In all this aforementioned work, both in the path integral formalism as well as in KL divergence control, the class of stochastic dynamical systems under consideration is rather restrictive since the

control transition matrix is state independent. Moreover, the connection to direct policy learning in RL and model-free learning was not made in any of the previous projects.

Our **PI**^{2} algorithm differs with respect to the aforementioned work in the following points.

- In Todorov (2009b) the stochastic optimal control problem is investigated for discrete action - state spaces and therefore it is treated as Markov Decision Process (MDP). To apply our **PI**^{2} algorithm, we do not discretize the state space and we do not treat the problem as an MDP. Instead we work in continuous state - action spaces which are suitable for performing RL in high dimensional robotic systems. To the best of our knowledge, our results present RL in one of the most high dimensional continuous state action spaces.

- In our derivations, the probabilistic interpretation of control comes directly from the Feynman-Kac Lemma. Thus we do not have to impose any artificial "pseudo-probability" treatment of the cost as in Todorov (2009b). In addition, for the continuous state - action spaces we do not have to learn the value function as it is suggested in Todorov (2009b) via Z-learning. Instead we directly find the controls based on our generalization of optimal controls.

- In the previous work, the problem of how to sample trajectories is not addressed. Sampling is performed at once with the hope to cover the all state space. We follow a rather different approach that allows to attack robotic learning problems of the complexity and dimensionality of the little dog robot.

- The work in Todorov (2009a) considers stochastic dynamics with state dependent control matrix. However, the way of how the stochastic optimal control problem is solved is by imposing strong assumptions on the structure of the cost function and, therefore, restrictions of the proposed solution to special cases of optimal control problems. The use of this specific cost function allows transforming the stochastic optimal control problem to a deterministic optimal control problem. Under this transformation, the stochastic optimal control problem can be solved by using deterministic algorithms.

- With respect to the work in Broek et al. (2008), Wiegerinck et al. (2006) and Kappen et al. (2009) our **PI**^{2} algorithm has been derived for a rather general class of systems with control transition matrix that is state dependent. In this general class, Rigid body and multi-body dynamics as well as the DMPs are included. Furthermore we have shown how our results generalize previous work.

4.2 Reinforcement Learning of Parameterized Policies

There are two main classes of related algorithms: Policy Gradient algorithms and probabilistic algorithms.

Policy Gradient algorithms (Peters and Schaal, 2006a,b) compute the gradient of the cost function (24) at every iteration and the policy parameters are updated according to $\theta^{(new)} = \theta^{(old)} + \alpha\nabla_\theta J$. Some well-established algorithms, which we will also use for comparisons, are as follows (see also Peters and Schaal, 2006a,b).

4.2.1 REINFORCE

Williams (1992) introduced the episodic REINFORCE algorithm, which is derived from taking the derivative of (24) with respect to the policy parameters. This algorithm has rather slow convergence

due to a very noisy estimate of the policy gradient. It is also very sensitive to a reward baseline parameter b_k (see below). Recent work derived the optimal baseline for REINFORCE (cf. Peters and Schaal, 2008a), which improved the performance significantly. The episodic REINFORCE update equations are:

$$\begin{aligned} \nabla_{\theta_k} J &= E_{\boldsymbol{\tau}_0}\left[(R(\boldsymbol{\tau}_0) - b_k) \sum_{i=0}^{N-1} \nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i}) \right], \\ b_k &= \frac{E_{\boldsymbol{\tau}_0}\left[\left(\sum_{i=0}^{N-1} \nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i})\right)^2 R(\boldsymbol{\tau}_0) \right]}{E_{\boldsymbol{\tau}_0}\left[\left(\sum_{i=0}^{N-1} \nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i})\right)^2 \right]}, \end{aligned}$$

where k denotes the k-th coefficient of the parameter vector and $R(\boldsymbol{\tau}_0) = \frac{1}{N}\sum_{i=0}^{N-1} r_{t_i}$.

4.2.2 GPOMDP AND THE POLICY GRADIENT THEOREM ALGORITHM

In their GPOMDP algorithm, Baxter and Bartlett (2001) introduced several improvements over REINFORCE that made the gradient estimates more efficient. GPOMDP can also be derived from the policy gradient theorem (Sutton et al., 2000; Peters and Schaal, 2008a), and an optimal reward baseline can be added (cf. Peters and Schaal, 2008a). In our context, the GPOMDP learning algorithm can be written as:

$$\begin{aligned} \nabla_{\theta_k} J &= E_{\boldsymbol{\tau}_0}\left[\sum_{j=0}^{N-1} (r_{tj} - b_{t_j}^{(k)}) \sum_{i=0}^{j} (\nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i})) \right], \\ b_{t_i}^{(k)} &= \frac{E_{\boldsymbol{\tau}_0}\left[(\nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i}))^2 r_{ti} \right]}{E_{\boldsymbol{\tau}_0}\left[(\nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i}))^2 \right]}. \end{aligned}$$

4.2.3 THE EPISODIC NATURAL ACTOR CRITIC

One of the most efficient policy gradient algorithm was introduced in Peters and Schaal (2008b), called the Episodic Natural Actor Critic. In essence, the method uses the Fisher Information Matrix to project the REINFORCE gradient onto a more effective update direction, which is motivated by the theory of natural gradients by Amari (1999). The eNAC algorithm takes the form of:

$$\begin{aligned} \xi_{t_i,k} &= \begin{bmatrix} \nabla_{\theta_k} \ln p(\mathbf{a}_{t_i}|\mathbf{x}_{t_i}) \\ 1 \end{bmatrix}, \\ \begin{bmatrix} \nabla_\theta J \\ J_0 \end{bmatrix} &= E_{\boldsymbol{\tau}_0}\left[\sum_{i=0}^{N-1} \xi_{t_i,k} \xi_{t_i,k}^T \right]^{-1} E_{\boldsymbol{\tau}_0}\left[R(\boldsymbol{\tau}_0) \sum_{i=0}^{N-1} \xi_{t_i,k} \right], \end{aligned}$$

where J_0 is a constant offset term.

4.2.4 POWER

The PoWER algorithm (Koeber and Peters, 2008) is a probabilistic policy improvement method, not a gradient algorithm. It is derived from an Expectation-Maximization framework using probability

matching (Dayan and Hinton, 1997; Peters and Schaal, 2008c). Using the notation of this paper, the parameter update of PoWER becomes:

$$\delta\theta = E_{\tau_0}\left[\sum_{t=0}^{N-1} R_{t_i}\frac{\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T}{\mathbf{g}_{t_i}^T\mathbf{g}_{t_i}}\right]^{-1} E_{\tau_0}\left[\sum_{t_i=t_o}^{t_N} R_{t_i}\frac{\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T\varepsilon_t}{\mathbf{g}_{t_i}^T\mathbf{g}_{t_i}}\right],$$

where $R_{t_i} = \sum_{j=i}^{N-1} r_{t_j}$. If we set $\mathbf{R}^{-1} = c\ \mathbf{I}$ in the update (37) of $\mathbf{PI}^2$, and set $\frac{\mathbf{g}_{t_i}\mathbf{g}_{t_i}^T}{\mathbf{g}_{t_i}^T\mathbf{g}_{t_i}} = \mathbf{I}$ in the matrix inversion term of (39), the two algorithms look essentially identical. But it should be noted that the rewards r_{t_i} in PoWER need to behave like an improper probability, that is, be strictly positive and integrate to a constant number—this property can make the design of suitable cost functions more complicated. $\mathbf{PI}^2$, in contrast, uses exponentiated sum of reward terms, where the immediate reward can be arbitrary, and only the cost on the motor commands needs be quadratic. Our empirical evaluations revealed that, for cost functions that share the same optimum in the PoWER pseudo-probability formulation and the $\mathbf{PI}^2$ notation, both algorithms perform essentially identical, indicating that the matrix inversion term in PoWER may be unimportant for many systems. It should be noted that in Vlassis et al. (2009), PoWER was extended to the discounted infinite horizon case, where PoWER is the special case of a non-discounted finite horizon problem.

5. Evaluations

We evaluated $\mathbf{PI}^2$ in several synthetic examples in comparison with REINFORCE, GPOMDP, eNAC, and, when possible, PoWER. Except for PoWER, all algorithms are suitable for optimizing immediate reward functions of the kind $r_t = q_t + \mathbf{u}_t\mathbf{R}\mathbf{u}_t$. As mentioned above, PoWER requires that the immediate reward behaves like an improper probability. This property is incompatible with $r_t = q_t + \mathbf{u}_t\mathbf{R}\mathbf{u}_t$ and requires some special nonlinear transformations, which usually change the nature of the optimization problem, such that PoWER optimizes a different cost function. Thus, only one of the examples below has a compatible a cost function for all algorithms, including PoWER. In all examples below, exploration noise and, when applicable, learning rates, were tuned for every individual algorithms to achieve the best possible numerically stable performance. Exploration noise was only added to the maximally activated basis function in a motor primitive,[10] and the noise was kept constant for the entire time that this basis function had the highest activation—empirically, this tick helped improves the learning speed of all algorithms.

5.1 Learning Optimal Performance of a 1 DOF Reaching Task

The first evaluation considers learning optimal parameters for a 1 DOF DMP (cf. Equation 30). The immediate cost and terminal cost are, respectively:

$$r_t = 0.5 f_t^2 + 5000\ \theta^T\theta, \qquad \phi_{t_N} = 10000(\dot{y}_{t_N}^2 + 10(g - y_{t_N})^2)$$

with $y_{t_0} = 0$ and $g = 1$—we use *radians* as units motivated by our interest in robotics application, but we could also avoid units entirely. The interpretation of this cost is that we would like to reach the goal g with high accuracy while minimizing the acceleration of the movement and while keeping the parameter vector short. Each algorithm was run for 15 trials to compute a parameter update, and

10. That is, the noise vector in (25) has only one non-zero component.

a total of 1000 updates were performed. Note that 15 trials per update were chosen as the DMP had 10 basis functions, and the eNAC requires at least 11 trials to perform a numerically stable update due to its matrix inversion. The motor primitives were initialized to approximate a 5-th order polynomial as point-to-point movement (cf. Figure 1a,b), called a minimum-jerk trajectory in the motor control literature; the movement duration was 0.5 seconds, which is similar to normal human reaching movements. Gaussian noise of $N(0,0.1)$ was added to the initial parameters of the movement primitives in order to have different initial conditions for every run of the algorithms. The results are given in Figure 1. Figure 1a,b show the initial (before learning) trajectory generated by the DMP together with the learning results of the four different algorithms after learning—essentially, all algorithms achieve the same result such that all trajectories lie on top of each other. In Figure 1c, however, it can be seen that **PI**^{2} outperforms the gradient algorithms by an order of magnitude. Figure 1d illustrates learning curves for the same task as in Figure 1c, just that parameter updates are computed already after two roll-outs—the eNAC was excluded from this evaluation as it would be too heuristic to stabilize its ill-conditioned matrix inversion that results from such few roll-outs. **PI**^{2} continues to converge much faster than the other algorithms even in this special scenario. However, there are some noticeable fluctuation after convergence. This noise around the convergence baseline is caused by using only two noisy roll-outs to continue updating the parameters, which causes continuous parameter fluctuations around the optimal parameters. Annealing the exploration noise, or just adding the optimal trajectory from the previous parameter update as one of the roll-outs for the next parameter update can alleviate this issue—we do not illustrate such little "tricks" in this paper as they really only affect fine tuning of the algorithm.

5.2 Learning Optimal Performance of a 1 DOF Via-Point Task

The second evaluation was identical to the first evaluation, just that the cost function now forced the movement to pass through an intermediate via-point at $t = 300ms$. This evaluation is an abstract approximation of hitting a target, for example, as in playing tennis, and requires a significant change in how the movement is performed relative to the initial trajectory (Figure 2a). The cost function was

$$r_{300ms} = 100000000(G - y_{t_{300ms}})^2, \qquad \phi_{t_N} = 0$$

with $G = 0.25$. Only this single reward was given. For this cost function, the PoWER algorithm can be applied, too, with cost function $\tilde{r}_{300ms} = \exp(-1/\lambda \quad r_{300ms})$ and $\tilde{r}_{t_i} = 0$ otherwise. This transformed cost function has the same optimum as r_{300ms}. The resulting learning curves are given in Figure 2 and resemble the previous evaluation: **PI**^{2} outperforms the gradient algorithms by roughly an order of magnitude, while all the gradient algorithms have almost identical learning curves. As was expected from the similarity of the update equations, PoWER and **PI**^{2} have in this special case the same performance and are hardly distinguishable in Figure 2. Figure 2a demonstrates that all algorithms pass through the desired target G, but that there are remaining differences between the algorithms in how they approach the target G—these difference have a small numerical effect in the final cost (where **PI**^{2} and PoWER have the lowest cost), but these difference are hardly task relevant.

5.3 Learning Optimal Performance of a Multi-DOF Via-Point Task

A third evaluation examined the scalability of our algorithms to a high-dimensional and highly redundant learning problem. Again, the learning task was to pass through an intermediate target G,

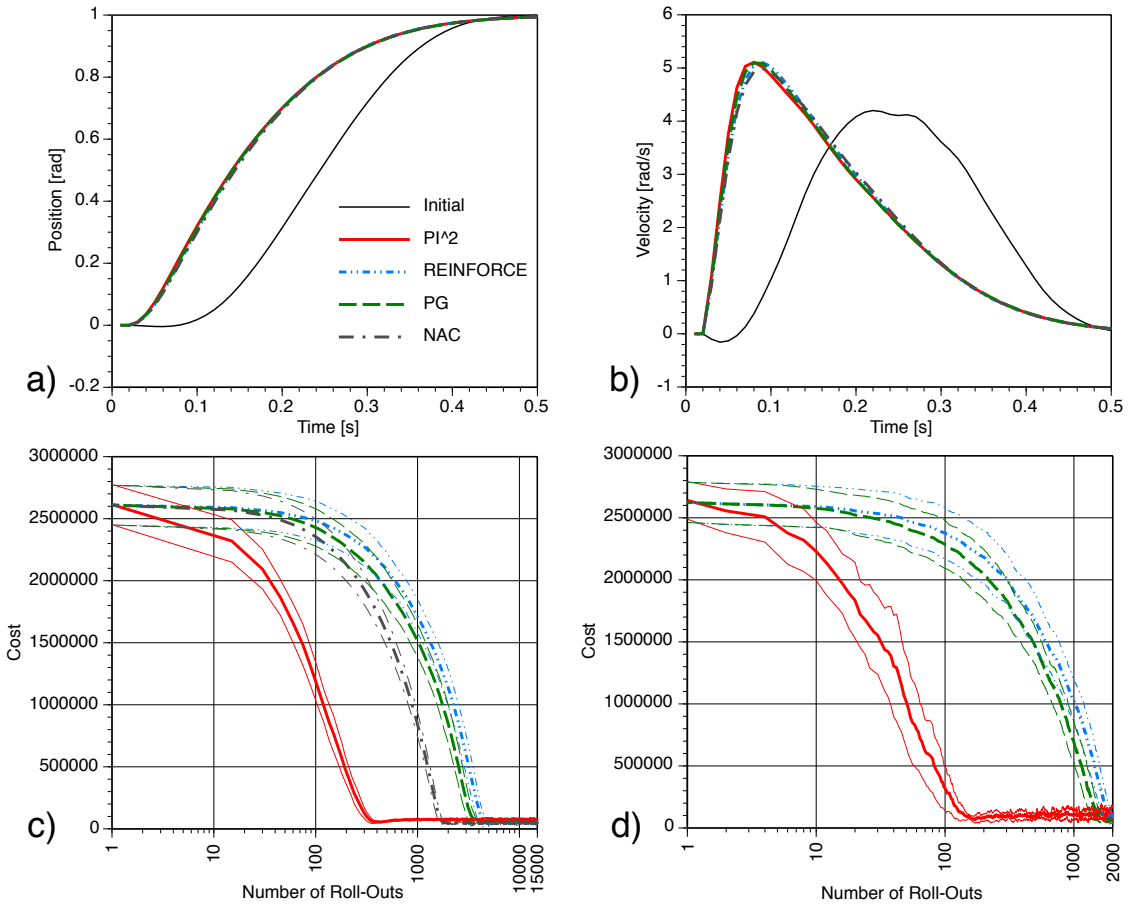


Figure 1: Comparison of reinforcement learning of an optimized movement with motor primitives. a) Position trajectories of the initial trajectory (before learning) and the results of all algorithms after learning—the different algorithms are essentially indistighuishable. b) The same as a), just using the velocity trajectories. c) Average learning curves for the different algorithms with 1 std error bars from averaging 10 runs for each of the algorithms. d) Learning curves for the different algorithms when only two roll-outs are used per update (note that the eNAC cannot work in this case and is omitted).

just that a $d = 2, 10$, or 50 dimensional motor primitive was employed. We assume that the multi-DOF systems model planar robot arms, where d links of equal length $l = 1/d$ are connected in an open chain with revolute joints. Essentially, these robots look like a multi-segment snake in a plane, where the tail of the snake is fixed at the origin of the 2D coordinate system, and the head of the snake can be moved in the 2D plane by changing the joint angles between all the links. Figure 3b,d,f illustrate the movement over time of these robots: the initial position of the robots is when all joint angles are zero and the robot arm completely coincides with the x-axis of the coordinate frame. The goal states of the motor primitives command each DOF to move to a joint angle, such that the entire robot configuration afterwards looks like a semi-circle where the most distal link of the robot

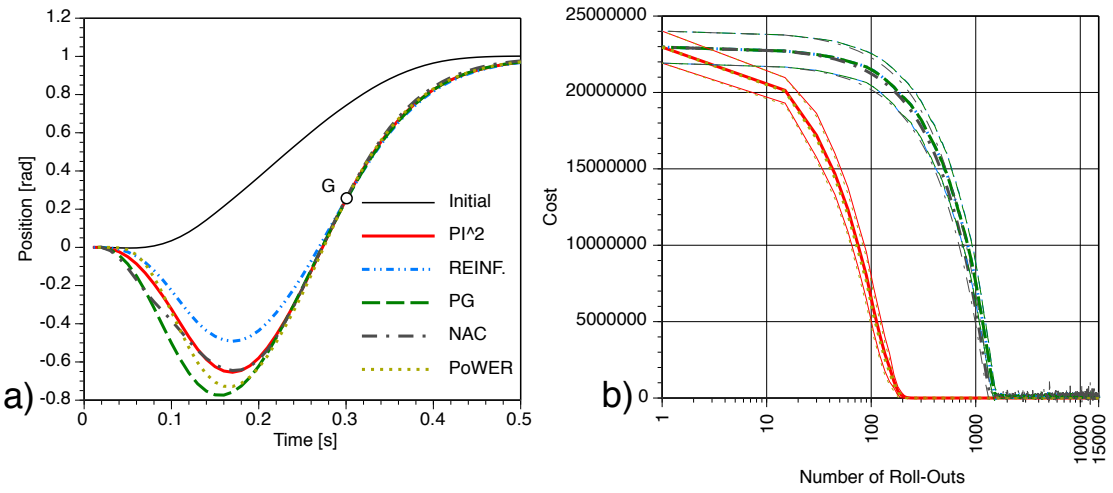


Figure 2: Comparison of reinforcement learning of an optimized movement with motor primitives for passing through an intermediate target G. a) Position trajectories of the initial trajectory (before learning) and the results of all algorithms after learning. b) Average learning curves for the different algorithms with 1 std error bars from averaging 10 runs for each of the algorithms.

(the end-effector) touches the y-axis. The higher priority task, however, is to move the end-effector through a via-point $G = (0.5, 0.5)$. To formalize this task as a reinforcement learning problem, we denote the joint angles of the robots as ξ_i, with $i = 1, 2, ..., d$, such that the first line of (30) reads now as $\ddot{\xi}_{i,t} = f_{i,t} + \mathbf{g}_{i,t}^T(\theta_i + \varepsilon_{i,t})$—this small change of notation is to avoid a clash of variables with the (x, y) task space of the robot. The end-effector position is computed as:

$$x_t = \frac{1}{d}\sum_{i=1}^{d} cos(\sum_{j=1}^{i} \xi_{j,t}), \quad y_t = \frac{1}{d}\sum_{i=1}^{d} sin(\sum_{j=1}^{i} \xi_{j,t}).$$

The immediate reward function for this problem is defined as

$$\begin{aligned} r_t &= \frac{\sum_{i=1}^{d}(d+1-i)\left(0.1 f_{i,t}^2 + 0.5\, \theta_i^T \theta_i\right)}{\sum_{i=1}^{d}(d+1-i)}, \\ \Delta r_{300ms} &= 100000000\left((0.5 - x_{t_{300ms}})^2 + (0.5 - y_{t_{300ms}})^2\right), \\ \phi_{t_N} &= 0, \end{aligned} \tag{39}$$

where Δr_{300ms} is added to r_t at time $t = 300ms$, that is, we would like to pass through the via-point at this time. The individual DOFs of the motor primitive were initialized as in the 1 DOF examples above. The cost term in (39) penalizes each DOF for using high accelerations and large parameter vectors, which is a critical component to achieve a good resolution of redundancy in the arm. Equation (39) also has a weighting term $d + 1 - i$ that penalizes DOFs proximal to the orgin more than those that are distal to the origin—intuitively, applied to human arm movements, this would mean that wrist movements are cheaper than shoulder movements, which is motivated by the

fact that the wrist has much lower mass and inertia and is thus energetically more efficient to move.

The results of this experiment are summarized in Figure 3. The learning curves in the left column demonstrate again that **PI**^{2} has an order of magnitude faster learning performance than the other algorithms, irrespective of the dimensionality. **PI**^{2} also converges to the lowest cost in all examples:

Algorithm	2-DOFs	10-DOFs	50-DOFs
PI^2	98000 ± 5000	15700 ± 1300	2800 ± 150
REINFORCE	125000 ± 2000	22000 ± 700	19500 ± 24000
PG	128000 ± 2000	28000 ± 23000	27000 ± 40000
NAC	113000 ± 10000	48000 ± 8000	22000 ± 2000

Figure 3 also illustrates the path taken by the end-effector before and after learning. All algorithms manage to pass through the via-point G appropriately, although the path particularly before reaching the via-point can be quite different across the algorithms. Given that **PI**^{2} reached the lowest cost with low variance in all examples, it appears to have found the best solution. We also added a "stroboscopic" sketch of the robot arm for the **PI**^{2} solution, which proceeds from the very right to the left as a function of time. It should be emphasized that there were absolutely no parameter tuning needed to achieve the **PI**^{2} results, while all gradient algorithms required readjusting of learning rates for every example to achieve best performance.

5.4 Application to Robot Learning

Figure 4 illustrates our application to a robot learning problem. The robot dog is to jump across as gap. The jump should make forward progress as much as possible, as it is a maneuver in a legged locomotion competition which scores the speed of the robot—note that we only used a physical simulator of the robot for this experiment, as the actual robot was not available. The robot has three DOFs per leg, and thus a total of $d = 12$ DOFs. Each DOF was represented as a DMP with 50 basis functions. An initial seed behavior (Figure 5-top) was taught by learning from demonstration, which allowed the robot barely to reach the other side of the gap without falling into the gap—the demonstration was generated from a manual adjustment of spline nodes in a spline-based trajectory plan for each leg.

PI^2 learning used primarily the forward progress as a reward, and slightly penalized the squared acceleration of each DOF, and the length of the parameter vector. Additionally, a penalty was incurred if the yaw or the roll exceeded a threshold value—these penalties encouraged the robot to

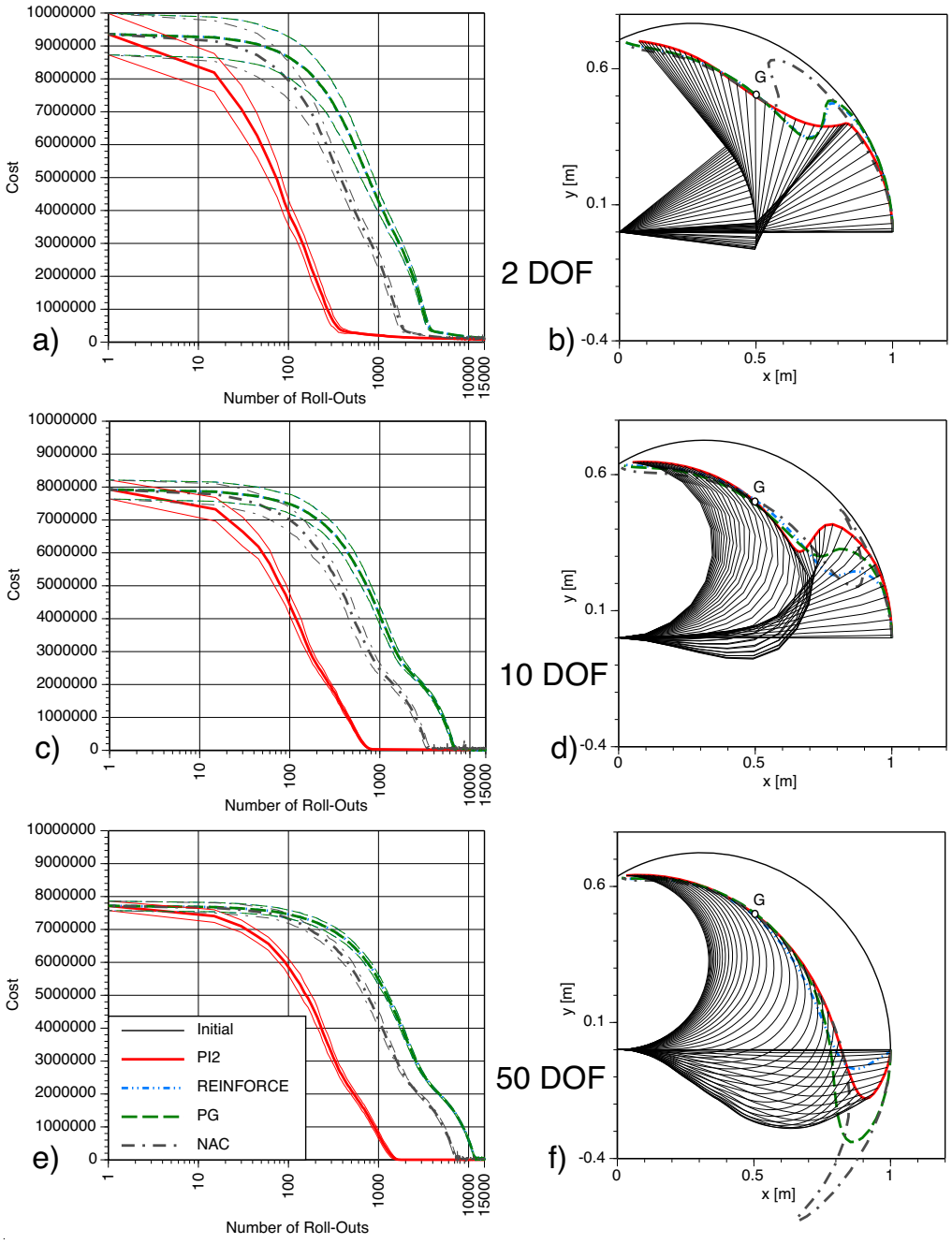
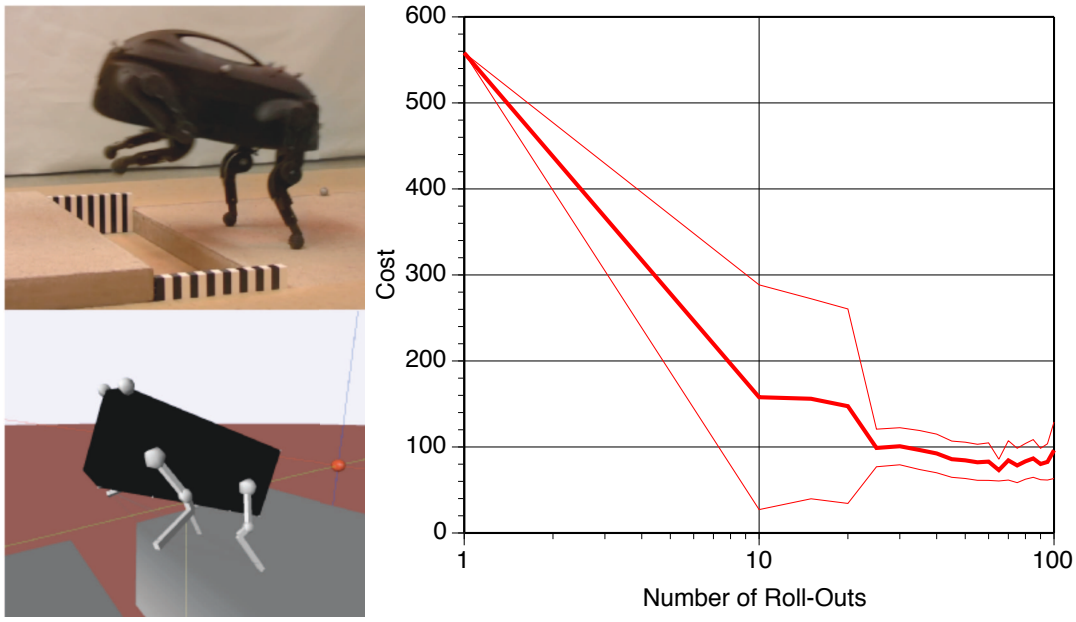


Figure 3: Comparison of learning multi-DOF movements (2,10, and 50 DOFs) with planar robot arms passing through a via-point G. a,c,e) illustrate the learning curves for different RL algorithms, while b,d,f) illustrate the end-effector movement after learning for all algorithms. Additionally, b,d,f) also show the initial end-effector movement, before learning to pass through G, and a "stroboscopic" visualization of the arm movement for the final result of $\mathbf{PI}^2$ (the movements proceed in time starting at the very right and ending by (almost) touching the y axis).



(a) Real & Simulated Robot Dog

(b) Learning curve for Dog Jump with $\mathbf{PI}^2$ $\pm 1 std$

Figure 4: Reinforcement learning of optimizing to jump over a gap with a robot dog. The improvement in cost corresponds to about 15 cm improvement in jump distance, which changed the robot's behavior from an initial barely successful jump to jump that completely traversed the gap with entire body. This learned behavior allowed the robot to traverse a gap at much higher speed in a competition on learning locomotion. The experiments for this paper were conducted only on the robot simulator.

jump straight forward and not to the side, and not to fall over. The exact cost function is:

$$
\begin{aligned}
r_t &= r_{roll} + r_{yaw} + \sum_{i=1}^{d} \left(a_1 f_{i,t}^2 + 0.5 a_2\, \theta_i^T \theta\right) (a_1 = 1.e-6, a_2 = 1.e-8), \\
r_{roll} &= \begin{cases} 100 * (|roll_t| - 0.3)^2, & \text{if } (|roll_t| > 0.3) \\ 0, & otherwise, \end{cases} \\
r_{yaw} &= \begin{cases} 100 * (|yaw_t| - 0.1)^2, & \text{if } (|yaw_t| > 0.1) \\ 0, & otherwise, \end{cases} \\
\phi_{t_N} &= 50000 (goal - x_{nose})^2,
\end{aligned}
$$

where $roll, yaw$ are the roll and yaw angles of the robot's body, and x_{nose} is the position of the front tip (the "nose") of the robot in the forward direction, which is the direction towards the $goal$. The multipliers for each reward component were tuned to have a balanced influence of all terms. Ten learning trials were performed initially for the first parameter update. The best 5 trials were kept, and five additional new trials were performed for the second and all subsequent updates. Essentially, this method performs importance sampling, as the rewards for the 5 trials in memory were re-computed

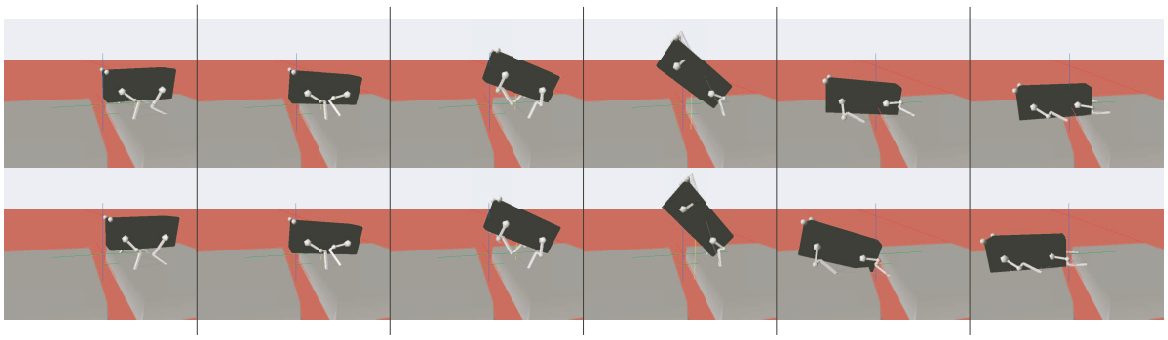


Figure 5: Sequence of images from the simulated robot dog jumping over a 14cm gap. Top: before learning. Bottom: After learning. While the two sequences look quite similar at the first glance, it is apparent that in the 4th frame, the robot's body is significantly higher in the air, such that after landing, the body of the dog made about 15cm more forward progress as before. In particular, the entire robot's body comes to rest on the other side of the gap, which allows for an easy transition to walking. In contrast, before learning, the robot's body (and its hind legs) are still on the right side of the gap, which does not allow for a successful continuation of walking.

with the latest parameter vectors. A total of 100 trials was performed per run, and ten runs were collected for computing mean and standard deviations of learning curves.

Figure 4 illustrates that after about 30 trials (i.e., 5 updates), the performance of the robot was converged and significantly improved, such that after the jump, almost the entire body was lying on the other side of the gap. Figure 4 captures the temporal performance in a sequence of snapshots of the robot. It should be noted that applying $\mathbf{PI}^2$ was algorithmically very simple, and manual tuning only focused on generated a good cost function, which is a different research topic beyond the scope of this paper.

6. Discussion

This paper derived a more general version of stochastic optimal control with path integrals, based on the original work by Kappen (2007) and Broek et al. (2008). The key results were presented in Table 1 and Section 2.5, which considered how to compute the optimal controls for a general class of stochastic control systems with state-dependent control transition matrix. One important class of these systems can be interpreted in the framework of reinforcement learning with parameterized policies. For this class, we derived Policy Improvement with Path Integrals ($\mathbf{PI}^2$) as a novel algorithm for learning a parameterized policy. $\mathbf{PI}^2$ inherits its sound foundation in first order principles of stochastic optimal control from the path integral formalism. It is a probabilistic learning method without open algorithmic tuning parameters, except for the exploration noise. In our evaluations, $\mathbf{PI}^2$ outperformed gradient algorithms significantly. It is also numerically simpler and has easier cost function design than previous probabilistic RL methods that require that immediate rewards are pseudo-probabilities. The similarity of $\mathbf{PI}^2$ with algorithms based on probability matching indicates that the principle of probability matching seems to approximate a stochastic optimal control framework. Our evaluations demonstrated that $\mathbf{PI}^2$ can scale to high dimensional control systems, unlike many other reinforcement learning systems.

Some issues, however, deserve more detailed discussions in the following paragraphs.

6.1 The Simplification $\lambda \mathbf{R}^{-1} = \Sigma_\varepsilon$

In order to obtain linear 2^{nd} order differential equations for the exponentially transformed HJB equations, the simplification $\lambda \mathbf{R}^{-1} = \Sigma_\varepsilon$ was applied. Essentially, this assumption couples the control cost to the stochasticity of the system dynamics, that is, a control with high variance will have relatively small cost, while a control with low variance will have relatively high cost. This assumption makes intuitively sense as it would be mostly unreasonable to attribute a lot of cost to an unreliable control component. Algorithmically, this assumption transforms the Gaussian probability for state transitions into a quadratic command cost, which is exactly what our immediate reward function postulated. Future work may allow removing this simplification by applying generalized versions of the Feynman-Kac Lemma.

6.2 Model-based, Hybrid, and Model-free Learning

Stochastic optimal control with path integrals makes a strong link to the dynamic system to be optimized—indeed, originally, it was derived solely as model-based method. As this paper demonstrated, however, this view can be relaxed. The roll-outs, needed for computing the optimal controls, can be generated either from simulating a model, or by gathering experience from an actual system. In the latter case, only the control transition matrix of the model needs be known, such that we obtain a hybrid model-based/model-free method. In this paper, we even went further and interpreted the stochastic dynamic system as a parameterized control policy, such that no knowledge of the model of the control system was needed anymore—that is, we entered a model-free learning domain. It seems that there is a rich variety of ways how the path integral formalism can be used in different applications.

6.3 Rules of Cost Function Design

The cost functions allowed in our formulations can have arbitrary state cost, but need quadratic command cost. This is somewhat restrictive, although the user can be flexible in what is defined as a command. For instance, the dynamic movement primitives (30) used in this paper can be written in two alternative ways:

$$\begin{aligned} \frac{1}{\tau}\dot{z}_t &= f_t + \mathbf{g}_t^T(\theta + \varepsilon_t), \\ &\text{or} \\ \frac{1}{\tau}\dot{z}_t &= \left[\mathbf{g}_t^T f_t\right]\left(\begin{bmatrix} \theta \\ 1 \end{bmatrix} + \tilde{\varepsilon}_t\right), \end{aligned}$$

where the new noise vector $\tilde{\varepsilon}_t$ has one additional coefficient. The second equation treats f_t as another basis function whose parameter is constant and is thus simply not updated. Thus, we added f_t to the command cost instead of treating it as a state cost.

We also numerically experimented with violations of the clean distinction between state and command cost. Equation (36) could be replaced by a cost term, which is an arbitrary function of state and command. In the end, this cost term is just used to differentiate the different roll-outs in a reward weighted average, similarly as in Peters and Schaal (2008c) and Koeber and Peters (2008). We noticed in several instances that $\mathbf{PI}^2$ continued to work just fine with this improper cost formulation.

Again, it appears that the path integral formalism and the **PI**^{2} algorithm allow the user to exploit creativity in designing cost functions, without absolute need to adhere perfectly to the theoretical framework.

6.4 Dealing with Hidden State

Finally, it is interesting to consider in how far **PI**^{2} would be affected by hidden state. Hidden state can either be of stochastic or deterministic nature, and we consider hidden state as adding additional equations to the system dynamics (3). Section 2.3 already derived that deterministic hidden states drop out of the **PI**^{2} update equations—these states of the system dynamics were termed as " non-directly actuated" states.

More interesting are hidden state variables that have stochastic differential equations, that is, these equations are uncontrolled but do have a noise term and a non-zero corresponding coefficient in $\mathbf{G}_t$ in Equation (3), and these equations are coupled to the other equations through their passive dynamics. The noise term of these equations would, in theory, contribute terms in Equation (36), but given that neither the noise nor the state of these equations are observable, we will not have the knowledge to add these terms. However, as long as the magnitude of these terms is small relative to the other terms in Equation (36), **PI**^{2} will continue to work fine, just a bit sub-optimally. This issue would affect other reinforcement learning methods for parameterized policies in the same way, and is not specific to **PI**^{2}.

6.5 Arbitrary States in the Cost Function

As a last point, we would like to consider which variables can actually enter the cost functions for **PI**^{2}. The path integral approach prescribes that the cost function needs to be a function of the state and command variables of the system equations (3). It should be emphasized that the state cost q_t can be any deterministic function of the state, that is, anything that is predictable from knowing the state, even if we do not know the predictive function. There is a lot of flexibility in this formulation, but it is also more restrictive than other approaches, for example, like policy gradients or the PoWER algorithm, where arbitrary variables can be used in the cost, no matter whether they are states or not.

We can think of any variable that we would like to use in the cost as having a corresponding differential equation in the system dynamics (3), that is, we simply add these variables as state variables, just that we do not know the analytical form of these equations. As in the previous section, it is useful to distinguish whether these states have deterministic or stochastic differential equations.

If the differential equation is deterministic, we can cover the case with the derivations from Section 2.3, that is, we consider such an equation as uncontrolled deterministic differential equation in the system dynamics, and we already know that we can use its state in the cost without any problems as it does not contribute to the probability of a roll-out.

If the differential equation is stochastic, the same argument as in the previous section applies, that is, the (unknown) contribution of the noise term of this equation to the exponentiated cost (36) needs to be small enough for **PI**^{2} to work effectively. Future work and empirical evaluations will have to demonstrate when these issues really matter—so far, we have not encountered problems in this regard.

7. Conclusions

The path integral formalism for stochastic optimal control has a very interesting potential to discover new learning algorithms for reinforcement learning. The $\mathbf{PI}^2$ algorithm derived in this paper for learning with parameterized policies demonstrated a surprisingly good performance, literally without any need for manual tuning of the parameters of the algorithm. We also demonstrated that the algorithm scales well into very high dimensional domains that were previously hardly approachable for reinforcement learning. Future work will thus allow us to focus much more on machine learning algorithms for cost function design, as the algorithmic components of the learning algorithm seem to be able to move towards a "black box" character.

Acknowledgments

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, IIS-9988642, ECS-0326095, ANI-0224419, the DARPA program on Learning Locomotion, the Multidisciplinary Research Program of the Department of Defense (MURI N00014-00-1-0637), and the ATR Computational Neuroscience Laboratories. J.B. was supported by a prospective researcher fellowship from the Swiss National Science Foundation. E.T. was supported by a Myronis Fellowship.

Appendix A.

Appendix **A** contains the lemmas **A1** and **A2** and one theorem. The theorem provides the main result of the generalized path integral control formalism expressed by (18), (19), (20). Its proof is based on results proven in the lemmas **A1** and **A2**. In appendix **B** we provide the Feynman-Kac formula and we sketch the corresponding proof.

Lemma 1 *: The optimal control solution to the stochastic optimal control problem expressed by (1),(2),(3) and (4) is formulated as:*

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[-\mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)T} \int \tilde{p}(\boldsymbol{\tau}_i) \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\boldsymbol{\tau}_i) d\boldsymbol{\tau}_i \right]$$

where $\tilde{p}(\boldsymbol{\tau}_i) = \frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)\right)}{\int \exp\left(-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)\right) d\boldsymbol{\tau}_i}$ *is a path dependent probability distribution. The term* $\tilde{S}(\boldsymbol{\tau}_i)$ *is a path function defined as* $\tilde{S}(\boldsymbol{\tau}_i) = S(\boldsymbol{\tau}_i) + \frac{\lambda}{2} \sum_{j=i}^{N-1} \log |\mathbf{H}_{t_j}|$ *that satisfies the following condition* $\lim_{dt \to 0} \int \exp\left(-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)\right) d\boldsymbol{\tau}_i \in \mathcal{C}^{(1)}$ *for any sampled trajectory starting from state* $\mathbf{x}_{t_i}$*. Moreover the term* $\mathbf{H}_{t_j}$ *is given by* $\mathbf{H}_{t_j} = \mathbf{G}_{t_j}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_j}^{(c)T}$ *while the term* $S(\boldsymbol{\tau}_i)$ *is defined according to*

$$S(\boldsymbol{\tau}_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2} \sum_{j=i}^{N-1} \| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \|_{\mathbf{H}_{t_j}}^2 dt.$$

Proof The optimal controls at the state $\mathbf{x}_{t_i}$ is expressed by the equation $\mathbf{u}_{t_i} = -\mathbf{R}^{-1} \mathbf{G}_{t_i} \nabla_{\mathbf{x}_{t_i}} V_{t_i}$. Due to the exponential transformation of the value function $\Psi_{t_i} = -\lambda \log V_{t_i}$ the equation of the optimal controls is written as:

$$\mathbf{u}_{t_i} = \lambda \mathbf{R}^{-1} \mathbf{G}_{t_i} \frac{\nabla_{\mathbf{x}_{t_i}} \Psi_{t_i}}{\Psi_{t_i}}.$$

In discrete time the optimal control is expressed as follows:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\nabla_{\mathbf{x}_{t_i}} \Psi_{t_i}^{(dt)}}{\Psi_{t_i}^{(dt)}} \right).$$

By using equation (17) and substituting $\Psi^{(dt)}(\mathbf{x}_{t_i}, t)$ we have:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\nabla_{\mathbf{x}_{t_i}} \int \exp\left(-\frac{1}{\lambda} Z(\tau_i)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} Z(\tau_i)\right) d\tau_i} \right).$$

Substitution of the term $Z(\tau_i)$ results in the equation:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\nabla_{\mathbf{x}_{t_i}} \int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i) - \frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i) - \frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) d\tau_i} \right).$$

Next we are using standard properties of the exponential function that lead to:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\nabla_{\mathbf{x}_{t_i}} \left[\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) \exp\left(-\frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) d\tau_i \right]}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) \exp\left(-\frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) d\tau_i} \right).$$

The term $\exp\left(-\frac{\lambda N l}{2} \log\left(2\pi dt \lambda\right)\right)$ does not depend on the trajectory τ_i, therefore it can be taken outside the integral as well as outside the gradient. Thus we will have that:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\exp\left(-\frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) \nabla_{\mathbf{x}_{t_i}} \left[\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i \right]}{\exp\left(-\frac{\lambda(N-i)l}{2} \log\left(2\pi dt \lambda\right)\right) \int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i} \right).$$

The constant term drops from the nominator and denominator and thus we can write:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \left[\frac{\nabla_{\mathbf{x}_{t_i}} \int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i} \right] \right).$$

Under the assumption that term $\exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i$ is continuously differentiable in $\mathbf{x}_{t_i}$ and dt we can change order of the integral with the differentiation operations. In general for $\nabla_x \int f(x,y) dy = \int \nabla_x f(x,y) dy$ to be true, $f(x,t)$ should be continuous in y and differentiable in x. Under this assumption, the optimal controls can be further formulated as:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\int \nabla_{\mathbf{x}_{t_i}} \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i} \right].$$

Application of the differentiation rule of the exponent results in:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \frac{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) \nabla_{\mathbf{x}_{t_i}} \left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i}{\int \exp\left(-\frac{1}{\lambda} \tilde{S}(\tau_i)\right) d\tau_i} \right].$$

The denominator is a function of $\mathbf{x}_{t_i}$ the current state and thus it can be pushed inside the integral of the nominator:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \int \frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)\right)}{\int \exp\left(-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)\right) d\boldsymbol{\tau}_i} \nabla_{\mathbf{x}_{t_i}} \left(-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i) \right) d\boldsymbol{\tau}_i \right].$$

By defining the probability $\tilde{p}(\boldsymbol{\tau}_i) = \frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)\right)}{\int \exp\left(-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i)\right) d\boldsymbol{\tau}_i}$ the expression above can be written as:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[\lambda \mathbf{R}^{-1} \mathbf{G}_{t_i}^T \int \tilde{p}(\boldsymbol{\tau}_i) \nabla_{\mathbf{x}_{t_i}} \left(-\frac{1}{\lambda}\tilde{S}(\boldsymbol{\tau}_i) \right) d\boldsymbol{\tau}_i \right].$$

Further simplification will result in:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[-\mathbf{R}^{-1} \mathbf{G}_{t_i}^T \int \tilde{p}(\boldsymbol{\tau}_i) \nabla_{\mathbf{x}_{t_i}} \tilde{S}(\boldsymbol{\tau}_i) d\boldsymbol{\tau}_i \right].$$

We know that the control transition matrix has the form $\mathbf{G}(\mathbf{x}_{t_i})^T = [0^T \quad \mathbf{G}_c(\mathbf{x}_{x_{t_i}})^T]$. In addition the partial derivative $\nabla_{\mathbf{x}_{t_i}} \tilde{S}(\boldsymbol{\tau}_i)$ can be written as $\nabla_{\mathbf{x}_{t_i}} \tilde{S}(\boldsymbol{\tau}_i)^T = [\nabla_{\mathbf{x}_{t_i}^{(m)}} \tilde{S}(\boldsymbol{\tau}_i)^T \quad \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\boldsymbol{\tau}_i)^T]$. By using these equations we will have that:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(-\mathbf{R}^{-1} [0^T \quad \mathbf{G}_{t_i}^{(c)T}] \int \tilde{p}(\boldsymbol{\tau}_o) \begin{bmatrix} \nabla_{\mathbf{x}_{t_i}^{(m)}} \tilde{S}(\boldsymbol{\tau}_i) \\ \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\boldsymbol{\tau}_i) \end{bmatrix} d\boldsymbol{\tau}_i \right).$$

The equation above can be written in the form:

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(-[0^T \quad \mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)T}] \int \tilde{p}(\boldsymbol{\tau}_i) \begin{bmatrix} \nabla_{\mathbf{x}_{t_i}^{(m)}} \tilde{S}(\boldsymbol{\tau}_i) \\ \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\boldsymbol{\tau}_i) \end{bmatrix} d\boldsymbol{\tau}_i \right).$$

or

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left(-[0^T \quad \mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)T}] \begin{bmatrix} \int \tilde{p}(\boldsymbol{\tau}_i) \cdot \nabla_{\mathbf{x}_{t_i}^{(m)}} \tilde{S}(\boldsymbol{\tau}_i) d\boldsymbol{\tau}_i \\ \int \tilde{p}(\boldsymbol{\tau}_i) \cdot \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\boldsymbol{\tau}_i) d\boldsymbol{\tau}_i \end{bmatrix} \right).$$

Therefore we will have the result

$$\mathbf{u}_{t_i} = \lim_{dt \to 0} \left[-\mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)T} \int \tilde{p}(\boldsymbol{\tau}_i) \nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\boldsymbol{\tau}_i) d\boldsymbol{\tau}_i \right].$$

■

Lemma 2 *: Given the stochastic dynamics and the cost in (1),(2),(3) and(4) the gradient of the path function $\tilde{S}(\boldsymbol{\tau}_i)$ with respect to the directly actuated part of the state $\mathbf{x}_{t_i}^{(c)}$ is formulated as:*

$$\lim_{dt \to 0} \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \tilde{S}(\boldsymbol{\tau}_i) \right) = -\mathbf{H}_{t_i}^{-1} \left(\mathbf{G}_{t_i}^{(c)} \boldsymbol{\varepsilon}_{t_i} - \mathbf{b}_{t_i} \right)$$

where the function $b(\mathbf{x}_{t_i})$ *defined as* $\lambda \mathbf{H}(\mathbf{x}_{t_i})\Phi_{t_i}$ *with* $\mathbf{H}_{t_i} = \mathbf{G}_{t_i}^{(c)} \mathbf{R}^{-1} \mathbf{G}_{t_i}^{(c)^T}$ *and the quantity* $\Phi_{t_i} \in \Re^{l \times 1}$ *is expressed as:*

$$\Phi_{t_i} = \frac{1}{2} \begin{bmatrix} trace\left(\mathbf{H}_{t_i}^{-1} \ \partial_{\mathbf{x}_{t_i}^{(c1)}} \mathbf{H}_{t_i}\right) \\ trace\left(\mathbf{H}_{t_i}^{-1} \ \partial_{\mathbf{x}_{t_i}^{(c2)}} \mathbf{H}_{t_i}\right) \\ \cdot \\ \cdot \\ \cdot \\ trace\left(\mathbf{H}_{t_i}^{-1} \ \partial_{\mathbf{x}_{t_i}^{(cl)}} \mathbf{H}_{t_i}\right) \end{bmatrix}.$$

Proof

We are calculating the term $\nabla_{\mathbf{x}_{t_o}^{(c)}} \tilde{S}(\tau_o)$. More precisely we have shown that

$$\tilde{S}(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2} \sum_{j=i}^{N-1} \| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \|_{\mathbf{H}_{t_j}}^2 dt + \frac{\lambda}{2} \sum_{j=i}^{N-1} \log |\mathbf{H}_{t_j}|.$$

To limit the length of our derivation we introduce the notation $\gamma_{t_j} = \alpha_{t_j}^T \ \mathbf{h}_{t_j}^{-1} \ \alpha_{t_j}$ and $\alpha_{t_j} = \left(\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)} - \mathbf{f}_{t_j}^{(c)} dt\right)$ and it is easy to show that $\| \frac{\mathbf{x}_{t_{j+1}}^{(c)} - \mathbf{x}_{t_j}^{(c)}}{dt} - \mathbf{f}_{t_j}^{(c)} \|_{\mathbf{H}_{t_j}}^2 dt = \frac{1}{dt} \gamma_{t_j}$ and therefore we will have:

$$\tilde{S}(\tau_i) = \phi_{t_N} + \frac{1}{2dt} \sum_{j=i}^{N-1} \gamma_{t_j} + \sum_{t_o}^{t_N} Q_{t_j} dt + \frac{\lambda}{2} \sum_{j=i}^{N-1} \log |\mathbf{H}_{t_j}|.$$

■

In the analysis that follows we provide the derivative of the 1th, 2th and 4th term of the cost function. We assume that the cost of the state during the time horizon $Q_{t_i} = 0$. In cases that this is not true then the derivative $\nabla_{\mathbf{x}_{t_i}^{(c)}} \sum_{t_i}^{t_N} Q_{t_i} dt$ needs to be found as well. By calculating the term $\nabla_{\mathbf{x}_{t_o}^{(c)}} \tilde{S}(\tau_o)$ we can find the local controls $\mathbf{u}(\tau_i)$. It is important to mention that the derivative of the path cost $S(\tau_i)$ is taken only with respect to the current state $\mathbf{x}_{t_o}$.

The first term is:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}} (\phi_{t_N}) = 0.$$

DERIVATIVE OF THE 2TH TERM $\nabla_{\mathbf{x}_{t_i}^{(c)}} \left[\frac{1}{2dt} \sum_{i=1}^{N-1} \gamma_{t_i}\right]$ OF THE COST $S(\tau_i)$.

The second term can be found as follows:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}} \left[\frac{1}{2dt} \sum_{j=i}^{N-1} \gamma_{t_j}\right].$$

The operator $\nabla_{\mathbf{x}_{t_o}^{(c)}}$ is linear and it can massaged inside the sum:

$$\frac{1}{2dt}\sum_{j=i}^{N-1}\nabla_{\mathbf{x}_{t_j}^{(c)}}\left(\gamma_{t_j}\right).$$

Terms that do not depend on $\mathbf{x}_{t_i}^{(c)}$ drop and thus we will have:

$$\frac{1}{2dt}\nabla_{\mathbf{x}_{t_i}^{(c)}}\gamma_{t_i}.$$

Substitution of the parameter $\gamma_{t_i}=\alpha_{t_i}^T\ \mathbf{H}_{t_i}^{-1}\ \alpha_{t_i}$ will result in:

$$\frac{1}{2dt}\nabla_{\mathbf{x}_{t_i}^{(c)}}\left[\alpha_{t_i}^T\ \mathbf{H}_{t_i}^{-1}\ \alpha_{t_i}\right].$$

By making the substitution $\beta_{t_i}=\mathbf{H}_{t_i}^{-1}\ \alpha_{t_i}$ and applying the rule $\nabla\left(\mathbf{u}(\mathbf{x})^T\mathbf{v}(\mathbf{x})\right)=\nabla\left(\mathbf{u}(\mathbf{x})\right)\mathbf{v}(\mathbf{x})+\nabla\left(\mathbf{v}(\mathbf{x})\right)\mathbf{u}(\mathbf{x})$ we will have that:

$$\frac{1}{2dt}\left[\nabla_{\mathbf{x}_{t_i}^{(c)}}\alpha_{t_i}\ \beta_{t_i}+\nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}\right]. \tag{40}$$

Next we find the derivative of α_{t_o}:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}}\alpha_{t_i}=\nabla_{\mathbf{x}_{t_i}^{(c)}}\left[\mathbf{x}_{t_{i+1}}^{(c)}-\mathbf{x}_{t_i}^{(c)}-\mathbf{f}_c(\mathbf{x}_{t_i})dt\right].$$

and the result is

$$\nabla_{\mathbf{x}_{t_i}^{(c)}}\alpha_{t_i}=-I_{l\times l}-\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}dt.$$

We substitute back to (40) and we will have:

$$\frac{1}{2dt}\left[-\left(I_{l\times l}+\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}dt\right)\ \beta_{t_i}+\nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}\right].$$

$$-\frac{1}{2dt}\left(I_{l\times l}+\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}dt\right)\ \beta_{t_i}+\frac{1}{2dt}\ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}.$$

After some algebra the result of $\nabla_{\mathbf{x}_{t_i}^{(c)}}\left(\frac{1}{2dt}\sum_{i=1}^{N-1}\gamma_{t_i}\right)$ is expressed as:

$$-\frac{1}{2dt}\ \beta_{t_i}-\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \beta_{t_i}+\frac{1}{2dt}\ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}.$$

The next step now is to find the limit of the expression above as $dt\to 0$. More precisely we will have that:

$$\lim_{dt\to 0}\left[-\frac{1}{2dt}\ \beta_{t_i}-\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \beta_{t_i}+\frac{1}{2dt}\ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \alpha_{t_i}\right].$$

LIMIT OF THE FIRST SUBTERM: $-\frac{1}{2dt}\ \beta_{t_i}$

We will continue our analysis by finding the limit for each one of the 3 terms above. The limit of the first term is calculated as follows:

$$\begin{aligned}\lim_{dt\to 0}\left(-\frac{1}{2dt}\ \ \beta_{t_i}\right) &= -\lim_{dt\to 0}\left(\frac{1}{2dt}\ \ \mathbf{H}_{t_i}^{-1}\ \ \alpha_{t_i}\right)\\ &= -\frac{1}{2}\ \ \mathbf{H}_{t_i}^{-1}\ \ \lim_{dt\to 0}\alpha_{t_i}\\ &= -\frac{1}{2}\ \ \mathbf{H}_{t_i}^{-1}\ \ \lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)}-\mathbf{x}_{t_i}^{(c)})\frac{1}{dt}-\mathbf{f}_{t_i}^{(c)}\right).\end{aligned}$$

LIMIT OF THE SECOND SUBTERM: $-\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \ \beta_{t_i}$

The limit of the second term is calculated as follows:

$$\begin{aligned}-\lim_{dt\to 0}\left(\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \ \beta_{t_i}\right) &= -\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_c(\mathbf{x}_{t_i})\ \ \lim_{dt\to 0}\beta_{t_i}\\ &= -\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_{t_i}^{(c)}\ \ \lim_{dt\to 0}\left(\mathbf{H}_{t_i}^{-1}\ \ \alpha_{t_i}\right)\\ &= -\frac{1}{2}\nabla_{\mathbf{x}_{t_i}^{(c)}}\mathbf{f}_c(\mathbf{x}_{t_i})\ \ \mathbf{H}_{t_i}^{-1}\ \ \lim_{dt\to 0}\alpha_{t_i}\\ &= 0.\end{aligned}$$

The limit of the term $\lim_{dt\to 0}\alpha_{t_i}$ is derived as:

$$\lim_{dt\to 0}\left(\mathbf{x}_{t_{i+1}}^{(c)}-\mathbf{x}_{t_i}^{(c)}-\mathbf{f}_c(\mathbf{x}_{t_i})dt\right)=\lim_{dt\to 0}\left(\mathbf{x}_{t_i+dt}^{(c)}-\mathbf{x}_{t_i}^{(c)}\right)-\lim_{dt\to 0}\mathbf{f}_c(\mathbf{x}_{t_i})dt=0-0=0.$$

LIMIT OF THE THIRD SUBTERM: $\frac{1}{2dt}\ \ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \ \alpha_{t_i}$

Finally the limit of the third term can be found as:

$$\begin{aligned}&\lim_{dt\to 0}\left(\frac{1}{2dt}\ \ \nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \ \alpha_{t_i}\right)=\\ &=\lim_{dt\to 0}\nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \ \lim_{dt\to 0}\left(\frac{1}{2dt}\ \ \alpha_{t_i}\right)=\\ &=\lim_{dt\to 0}\nabla_{\mathbf{x}_{t_i}^{(c)}}\beta_{t_i}\ \ \frac{1}{2}\lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)}-\mathbf{x}_{t_i}^{(c)})\frac{1}{dt}-\mathbf{f}_{t_i}^{(c)}\right).\end{aligned}$$

We substitute $\beta_{t_i}=\mathbf{H}_{t_i}^{-1}\ \ \alpha_{t_i}$ and write the matrix $\mathbf{H}_{t_i}^{-1}$ in row form:

$$= \lim_{dt\to 0} \nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\mathbf{H}_{t_i}^{-1} \;\; \alpha_{t_i}\right) \;\; \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_i}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right) =$$

$$= \lim_{dt\to 0} \nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\begin{bmatrix} \mathbf{H}_{t_i}^{(1)^{-T}} \\ \mathbf{H}_{t_i}^{(2)^{-T}} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{H}_{t_i}^{(l)^{-T}} \end{bmatrix} \;\; \alpha_{t_i} \right) \;\; \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right)$$

$$= \lim_{dt\to 0} \nabla_{\mathbf{x}_{t_i}^{(c)}} \begin{bmatrix} \mathbf{H}_{t_i}^{(1)^{-T}} & \alpha_{t_i} \\ \mathbf{H}_{t_i}^{(2)^{-T}} & \alpha_{t_i} \\ \cdot & \\ \cdot & \\ \cdot & \\ \mathbf{H}_{t_i}^{(l)^{-T}} & \alpha_{t_i} \end{bmatrix} \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right).$$

We can push the operator $\nabla_{\mathbf{x}_{t_i}^{(c)}}$ insight the matrix and apply it to each element.

$$= \lim_{dt\to 0} \begin{bmatrix} \nabla^T_{\mathbf{x}_{t_i}^{(c)}} \left(\mathbf{H}_{t_i}^{(1)^{-T}} \;\; \alpha_{t_i} \right) \\ \nabla^T_{\mathbf{x}_{t_i}^{(c)}} \left(\mathbf{H}_{t_i}^{(2)^{-T}} \;\; \alpha_{t_i} \right) \\ \cdot \\ \cdot \\ \cdot \\ \nabla^T_{\mathbf{x}_{t_i}^{(c)}} \left(\mathbf{H}_{t_i}^{(l)^{-T}} \;\; \alpha_{t_i} \right) \end{bmatrix} \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right).$$

We again use the rule $\nabla \left(\mathbf{u}(\mathbf{x})^T \mathbf{v}(\mathbf{x}) \right) = \nabla \left(\mathbf{u}(\mathbf{x}) \right) \mathbf{v}(\mathbf{x}) + \nabla \left(\mathbf{v}(\mathbf{x}) \right) \mathbf{u}(\mathbf{x})$ and thus we will have:

$$= \lim_{dt\to 0} \begin{bmatrix} \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(1)^{-T}} \;\; \alpha_{t_i} + \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i} \;\; \mathbf{H}_{t_i}^{(1)^{-T}} \right)^T \\ \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(2)^{-T}} \;\; \alpha_{t_i} + \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i} \;\; \mathbf{H}_{t_i}^{(2)^{-T}} \right)^T \\ \cdot \\ \cdot \\ \cdot \\ \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(l)^{-T}} \;\; \alpha_{t_i} + \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i} \;\; \mathbf{H}_{t_i}^{(l)^{-T}} \right)^T \end{bmatrix} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right).$$

We can split the matrix above into two terms and then we pull out the terms α_{t_i} and $\nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}$ respectively :

$$= \lim_{dt\to 0} \left(\alpha_{t_i}^T \begin{bmatrix} \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(1)^{-T}} \\ \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(2)^{-T}} \\ \cdot \\ \cdot \\ \cdot \\ \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{(l)^{-T}} \end{bmatrix} + \begin{bmatrix} \mathbf{H}_{t_i}^{(1)^{-T}} \\ \mathbf{H}_{t_i}^{(2)^{-T}} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{H}_{t_i}^{(l)^{-T}} \end{bmatrix} \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}^T \right) \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right)$$

$$= \lim_{dt\to 0} \left(\alpha_{t_i}^T \;\; \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{-1} + \mathbf{H}_{t_i}^{-1} \;\; \nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}^T \right) \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right)$$

$$= \left(\lim_{dt\to 0} \left(\alpha_{t_i}^T \right) \;\; \nabla_{\mathbf{x}_{t_i}^{(c)}} \mathbf{H}_{t_i}^{-1} + \mathbf{H}_{t_i}^{-1} \;\; \lim_{dt\to 0} \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}^T \right) \right) \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)} \right).$$

Since $\lim_{dt\to 0} \left(\alpha_{t_i}^T \right) = 0_{1\times l}$ and $\lim_{dt\to 0} \left(\nabla_{\mathbf{x}_{t_i}^{(c)}} \alpha_{t_i}^T \right) = -I_{l\times l}$ the final result is expressed as follows

$$\lim_{dt\to 0} \left(\frac{1}{2dt} \nabla_{\mathbf{x}_{t_i}^{(c)}} \beta_{t_i} \;\; \alpha_{t_i} \right) = -\mathbf{H}_{t_i}^{-1} \;\; \frac{1}{2} \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_o}^{(c)} \right).$$

After we have calculated the 3 sub-terms, the 2th term of the of the derivative of path cost $S(\tau_o)$ can be expressed in the following form:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\frac{1}{2\lambda dt} \sum_{j=i}^{N-1} \gamma_{t_j} \right) = -\mathbf{H}_{t_i}^{-1} \;\; \lim_{dt\to 0} \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)}) \frac{1}{dt} - \mathbf{f}_{t_i}^{(c)}) \right).$$

DERIVATIVE OF THE FOURTH TERM $\nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\frac{\lambda}{2} \sum_{j=i}^{N-1} \log |\mathbf{H}_{t_j}| \right)$ OF THE COST $S(\tau_i)$.

The analysis for the 4th term is given below:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}} \left(\frac{\lambda}{2} \sum_{j=i}^{N-1} \log |\mathbf{H}_{t_j}| \right) = \frac{\lambda}{2} \nabla_{\mathbf{x}_{t_i}^{(c)}} \log |\mathbf{H}_{t_i}|.$$

If we assume that $\mathbf{x}_{t_o}^{(c)} = [x_{t_o}^{(c1)}, x_{t_o}^{(c2)} .... x_{t_o}^{(cl)}]$ and take the derivative with respect to each element we will have

$$\partial_{\mathbf{x}_{t_i}^{(ci)}} \left(\frac{\lambda}{2} \log |\mathbf{H}_{t_i}| \right) = \frac{\lambda}{2} \frac{1}{|\mathbf{H}_{t_i}|} \;\; \partial_{\mathbf{x}_{t_i}^{(ci)}} |\mathbf{H}_{t_i}|.$$

$$\partial_{\mathbf{x}_{t_i}^{(ci)}} \left(\frac{\lambda}{2} \log |\mathbf{H}_{t_i}| \right) = \frac{\lambda}{2} \frac{1}{|\mathbf{H}(\mathbf{x}_{t_i})|} \;\; |\mathbf{H}_{t_i}| \;\; trace \left(\mathbf{H}_{t_i}^{-1} \cdot \partial_{\mathbf{x}_{t_i}^{(ci)}} \mathbf{H}_{t_i} \right).$$

$$\partial_{\mathbf{x}_{t_i}^{(ci)}} \left(\frac{\lambda}{2} \log |\mathbf{H}_{t_i}| \right) = \frac{\lambda}{2} trace \left(\mathbf{H}_{t_i}^{-1} \;\; \partial_{\mathbf{x}_{t_i}^{(ci)}} \mathbf{H}_{t_i} \right).$$

Where we make used of the identity $\partial det(A) = det(A)Tr\left(A^{-1}\partial A\right)$. The result is expressed as:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}}\left(\frac{\lambda}{2}\log|\mathbf{H}_{t_i}|\right) = \frac{\lambda}{2}\begin{bmatrix} trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(c1)}}\mathbf{H}_{t_i}\right) \\ trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(c2)}}\mathbf{H}_{t_i}\right) \\ \cdot \\ \cdot \\ \cdot \\ trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(cl)}}\mathbf{H}_{t_i}\right) \end{bmatrix}.$$

or in a more compact form:

$$\nabla_{\mathbf{x}_{t_i}^{(c)}}\left(\frac{\lambda}{2}\log|\mathbf{H}_{t_i}|\right) = \mathbf{H}_{t_i}^{-1}\mathbf{b}_{t_i}.$$

where $\mathbf{b}(\mathbf{x}_{t_i}) = \lambda\mathbf{H}(\mathbf{x}_{t_i})\Phi_{t_i}$ and the quantity $\Phi_{t_i} \in \Re^{l\times 1}$ is defined as:

$$\Phi_{t_i} = \frac{1}{2}\begin{bmatrix} trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(c1)}}\mathbf{H}_{t_i}\right) \\ trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(c2)}}\mathbf{H}_{t_i}\right) \\ \cdot \\ \cdot \\ \cdot \\ trace\left(\mathbf{H}_{t_i}^{-1}\ \partial_{\mathbf{x}_{t_i}^{(cl)}}\mathbf{H}_{t_i}\right) \end{bmatrix}. \tag{41}$$

Since we computed all the terms of the derivative of the path cost $\tilde{S}(\tau_o)$ and after putting all the terms together we have the result expressed as follows:

$$\lim_{dt\to 0}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right) = -\mathbf{H}_{t_i}^{-1}\left(\lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)})\frac{1}{dt} - \mathbf{f}_{t_i}^{(c)}\right) - \mathbf{b}_{t_i}\right).$$

By taking into account the fact that $\lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)})\frac{1}{dt} - \mathbf{f}_{t_i}^{(c)}\right) = \mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i}$ we get the following final expression:

$$\lim_{dt\to 0}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right) = -\mathbf{H}_{t_i}^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right).$$

Theorem 3 ***: The optimal control solution to the stochastic optimal control problem expressed by (1),(2),(3),(4) is formulated by the equation that follows:***

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\int \tilde{p}(\tau_i)\ \mathbf{u}_L(\tau_i)\ d\tau_i,$$

where $\tilde{p}(\tau_i) = \frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right)}{\int \exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_i)\right) d\tau_i}$ ***is a path depended probability distribution and the term*** $\mathbf{u}(\tau_i)$ ***defined as*** $\mathbf{u}_L(\tau_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\left(\mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\right)^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right)$ ***are the local controls of each sampled trajectory starting from state*** $\mathbf{x}_{t_i}$***. The terms*** ε_{t_i} ***and*** $\mathbf{b}_{t_i}$ ***are defined as*** $\varepsilon_{t_i} = \left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)})\frac{1}{dt} - \mathbf{f}_{t_i}^{(o)}\right)$ ***and*** $\mathbf{b}(\mathbf{x}_{t_i}) = \lambda\mathbf{H}(\mathbf{x}_{t_i})\Phi_{t_i}$ ***with*** $\mathbf{H}_{t_i} = \mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)^T}$ ***and*** Φ_{t_i} ***given in (41).***

Proof

To prove the theorem we make use of the Lemma 2 and we substitute $\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)$ in the main result of Lemma 1. More precisely from lemma A1 we have that:

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\left(\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\int \tilde{p}(\tau_i)\,\mathbf{H}_{t_i}^{-1}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right)d\tau_i\right).$$

The terms $\mathbf{R}^{-1}$ and $\mathbf{G}_{t_i}$ can be pushed insight the integral since they are independent of $\tau_i = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$. Thus we have the expression:

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\left(\int \tilde{p}(\tau_i)\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\left(\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i)\right)d\tau_i\right),$$

$$\mathbf{u}_{t_i} = \lim_{dt\to 0}\int \tilde{p}(\tau_i)\ \mathbf{u}_L^{(dt)}(\tau_i)\,d\tau_i,$$

where the local controls $\mathbf{u}_L^{(dt)}(\tau_i)$ are given as follows:

$$\mathbf{u}_L^{(dt)}(\tau_i) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\nabla_{\mathbf{x}_{t_i}^{(c)}}\tilde{S}(\tau_i).$$

After applying the limit, and making use of the result in Lemma 2 the equation above is expressed as:

$$\mathbf{u}_{t_i} = \int \tilde{p}(\tau_i)\,\mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i})d\tau_i,$$

where the local controls $\mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i})$ are given as follows:

$$\mathbf{u}_L(\tau_i) = \mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i}) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\left(\lim_{dt\to 0}\left((\mathbf{x}_{t_{i+1}}^{(c)} - \mathbf{x}_{t_i}^{(c)})\frac{1}{dt} - \mathbf{f}_{t_i}^{(c)}\right) - \mathbf{b}_{t_i}\right),$$

or in a simpler form:

$$\mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i}) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\mathbf{H}_{t_i}^{-1}\left(\mathbf{G}_c\varepsilon_{t_i} - \mathbf{b}_{t_i}\right).$$

By substituting with $\mathbf{H}(\mathbf{x}_{t_i}) = \mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}$ we have the final result:

$$\mathbf{u}_L(\tau_i) = \mathbf{u}_L(\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_i}) = \mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\left(\mathbf{G}_{t_i}^{(c)}\mathbf{R}^{-1}\mathbf{G}_{t_i}^{(c)T}\right)^{-1}\left(\mathbf{G}_{t_i}^{(c)}\varepsilon_{t_i} - \mathbf{b}_{t_i}\right).$$

■

Appendix B.

Theorem 4 *: Let us assume that* $\mathbf{x}$ *satisfies the SDE* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},t) + \mathbf{G}(\mathbf{x})\varepsilon(t)$. *Then* $\Psi(\mathbf{x},t_o) = \Psi(\mathbf{x},t_o,t_N) = E\left(\Psi(\mathbf{x},t_N)\, e^{\int_{t_o}^{t_N} -\frac{1}{\lambda} q(\mathbf{x}) d\tau}\right)$ *if and only if* $\Psi(\mathbf{x},t)$ *satisfies the backward Kolmogorov PDE:*

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + \mathbf{f}_t^T (\nabla_{\mathbf{x}} \Psi_t) + \frac{1}{2} trace\left((\nabla_{\mathbf{xx}} \Psi_t) \mathbf{G}_t \Sigma_\varepsilon \mathbf{G}_t^T\right),$$

with boundary condition:

$$\Psi(\mathbf{x},t_N) = \exp\left(-\frac{1}{\lambda}\phi(\mathbf{x}(t_N))\right).$$

Proof Given that $\mathbf{x}$ satisfies the SDE $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},t) + \mathbf{G}(\mathbf{x})\varepsilon(t)$ and $\Psi(\mathbf{x},t)$ satisfies the PDE above, application of Ito lemma (Øksendal, 2003) to function $Y(t) = \Psi(\mathbf{x}_t,t)\, e^{\int_{t_o}^{t} -\frac{1}{\lambda} q(\mathbf{x}) d\tau}$ leads to the final result $\Psi(\mathbf{x},t_o) = E\left(\Psi(\mathbf{x},t_N)\, e^{\int_{t_o}^{t_N} -\frac{1}{\lambda} q(\mathbf{x}) d\tau}\right)$. This result is the solution of the linear PDE. ■

References

S. Amari. Natural gradient learning for over- and under-complete bases in ica. *Neural Computation*, 11(8):1875–83, 1999.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5): 115–133, 1983.

J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

R. Bellman and R. Kalaba. *Selected Papers On mathematical trends in Control Theory*. Dover Publications, 1964.

B. Van Den Broek, W. Wiegerinck, and H. J. Kappen. Graphical model inference in optimal control of stochastic multi-agent systems. *Journal of Artificial Intelligence Research*, 32(1):95–122, 2008.

J. Buchli, E. Theodorou, F. Stulp, and S. Schaal. Variable impedance control - a reinforcement learning approach. In *Robotics Science and Systems*, 2010.

P. Dayan and G. Hinton. Using em for reinforcement learning. *Neural Computation*, 9, 1997.

M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian Process Dynamic Programming. *Neurocomputing*, 72(7–9):1508–1524, March 2009.

W. H. Fleming and H. M. Soner. *Controlled Markov Processes and Viscosity Solutions*. Applications of aathematics. Springer, New York, 2nd edition, 2006.

M. Ghavamzadeh and E. Yaakov. Bayesian actor-critic algorithms. In *ICML '07: Proceedings of The 24th International Conference on Machine Learning*, pages 297–304, 2007.

V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3:671–692, 1990.

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1547–1554. Cambridge, MA: MIT Press, 2003.

D. H. Jacobson and D. Q. Mayne. *Differential dynamic programming*. American Elsevier Pub. Co., New York,, 1970.

N. Jetchev and M Toussaint. Trajectory prediction: learning to map situations to robot trajectories. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 449–456, 2009.

H. J. Kappen. Linear theory for control of nonlinear stochastic systems. *Phys. Rev. Lett.*, 95:200201, Nov 2005a.

H. J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory and Experiment*, (11):P11011, 2005b.

H. J. Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. In J. Marro, P. L. Garrido, and J. J. Torres, editors, *Cooperative Behavior in Neural Systems*, volume 887 of *American Institute of Physics Conference Series*, pages 149–181, February 2007.

H. J. Kappen, W. Wiegerinck, and B. van den Broek. A path integral approach to agent planning. In *AAMAS*, 2007.

H. J. Kappen, Gmez V., and Opper M. Optimal control as a graphical model inference problem. *Journal for Machine Learning Research (JMLR)*, arXiv:0901.0633v, 2009. Submitted.

J. Koeber and J. Peters. Policy search for motor primitives. In D. Schuurmans, J. Benigio, and D. Koller, editors, *Advances in Neural Information Processing Systems 21 (NIPS 2008)*, pages 297–304, Vancouver, BC, Dec. 8-11, 2008. Cambridge, MA: MIT Press.

W. Thomas Miller, Richard S., and Paul J. Werbos. *Neural Networks for Control*. Neural network modeling and connectionism. MIT Press, Cambridge, Mass., 1990.

B. K. Øksendal. *Stochastic Differential Equations : An Introduction with Applications*. Springer, Berlin; New York, 6th edition, 2003.

J. Peters. *Machine Learning of Motor Skills for Robotics*. PhD thesis, University of Southern California, 2007.

J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE International Conference on Intelligent Robotics Systems (IROS 2006)*, 2006a.

J. Peters and S. Schaal. Reinforcement learning for parameterized motor primitives. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN 2006)*, 2006b.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–97, 2008a.

J. Peters and S. Schaal. Natural actor critic. *Neurocomputing*, 71(7-9):1180–1190, 2008b.

J. Peters and S. Schaal. Learning to control in operational space. *International Journal of Robotics Research*, 27:197–212, 2008c.

T. Rueckstiess, M. Felder, and J. Schmidhuber. State-dependent exploration for policy gradient methods. In *ECML PKDD '08: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pages 234–249, 2008.

S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Advanced textbooks in control and signal processing. Springer, London ; New York, 2000.

R. F. Stengel. *Optimal Control and Estimation*. Dover books on advanced mathematics. Dover Publications, New York, 1994.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, 1998.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.

E. Todorov. Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural Computation*, 17(5):1084, 2005.

E. Todorov. Linearly-solvable markov decision problems. In B. Scholkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS 2007)*, Vancouver, BC, 2007. Cambridge, MA: MIT Press.

E. Todorov. General duality between optimal control and estimation. In *In Proceedings of the 47th IEEE Conference on Decision and Control*, 2008.

E. Todorov. Classic maximum principles and estimation-control dualities for nonlinear stochastic systems. 2009a. (Submitted).

E. Todorov. Efficient computation of optimal actions. *Proceedings National Academy of Science USA*, 106(28):11478–83, 2009b.

M. Toussaint and A. Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes, 2006.

N. Vlassis, M. Toussaint, G. Kontes, and Piperidis. S. Learning model-free control by a monte-carlo em algorithm. *Autonomous Robots*, 27(2):123–130, 2009.

W. Wiegerinck, B. van den Broek, and H. J. Kappen. Stochastic optimal control in continuous space-time multi-agent system. In *UAI*, 2006.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

J. Yong. Relations among odes, pdes, fsdes, bsdes, and fbsdes. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 3, pages 2779–2784, Dec 1997.

Journal of Machine Learning Research 11 (2010) 3183-3234 Submitted 11/09; Revised 7/10; Published 11/10

A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification

Guo-Xun Yuan R96042@CSIE.NTU.EDU.TW
Kai-Wei Chang B92084@CSIE.NTU.EDU.TW
Cho-Jui Hsieh B92085@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Editor: Sathiya Keerthi

Abstract

Large-scale linear classification is widely used in many areas. The L1-regularized form can be applied for feature selection; however, its non-differentiability causes more difficulties in training. Although various optimization methods have been proposed in recent years, these have not yet been compared suitably. In this paper, we first broadly review existing methods. Then, we discuss state-of-the-art software packages in detail and propose two efficient implementations. Extensive comparisons indicate that carefully implemented coordinate descent methods are very suitable for training large document data.

Keywords: L1 regularization, linear classification, optimization methods, logistic regression, support vector machines, document classification

1. Introduction

Recently, L1-regularized classifiers have attracted considerable attention because they can be used to obtain a sparse model. Given a set of instance-label pairs $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, l$, $\mathbf{x}_i \in R^n$, $y_i \in \{-1, +1\}$, training an L1-regularized linear classifier involves the following unconstrained optimization problem:

$$\min_{\mathbf{w}} \quad f(\mathbf{w}) \equiv \|\mathbf{w}\|_1 + C \sum_{i=1}^{l} \xi(\mathbf{w}; \mathbf{x}_i, y_i), \tag{1}$$

where $\|\cdot\|_1$ denotes the 1-norm and $\xi(\mathbf{w}; \mathbf{x}_i, y_i)$ is a non-negative (convex) loss function. The regularization term $\|\mathbf{w}\|_1$ is used to avoid overfitting the training data. The user-defined parameter $C > 0$ is used to balance the regularization and loss terms.

If $\|\mathbf{w}\|_2$ is used in (1), we obtain an L2-regularized classifier. Although L2 regularization is used more commonly, an L1-regularized formula often produces a sparse $\mathbf{w}$. Nonzero elements help to select important features; in addition, the time required to produce predictions may be reduced. Considerable literature has been published on the advantages of using L1 regularization; see, for example, Zhao and Yu (2006). However, an L1-regularized form (1) is not differentiable regardless

of its loss function. This drawback leads to greater difficulty in solving the optimization problem. Therefore, certain considerations are required to handle the non-differentiability.

Many loss functions can be used for linear classification. A commonly used one is logistic loss:

$$\xi_{\log}(\mathbf{w};\mathbf{x},y) = \log(1+e^{-y\mathbf{w}^T\mathbf{x}}). \tag{2}$$

This loss function is twice differentiable. Note that minimizing logistic loss is equivalent to maximizing the likelihood, whereas minimizing the regularized loss in (1) is equivalent to maximizing the posterior with independent Laplace priors on the parameters. Two other frequently used functions are the L1- and the L2-loss functions:

$$\xi_{\mathrm{L1}}(\mathbf{w};\mathbf{x},y) = \max(1-y\mathbf{w}^T\mathbf{x},\ 0) \quad \text{and} \tag{3}$$

$$\xi_{\mathrm{L2}}(\mathbf{w};\mathbf{x},y) = \max(1-y\mathbf{w}^T\mathbf{x},\ 0)^2. \tag{4}$$

Because of the $\max(\cdot)$ operation, the L1-loss function is not differentiable. On the other hand, L2 loss is differentiable, but not twice differentiable (Mangasarian, 2002). We refer to (1) with logistic loss as L1-regularized logistic regression and (1) with L1/L2 loss as L1-regularized L1-/L2-loss support vector machines (SVMs).

In some applications, we require a bias term b (also called as an intercept) in the loss function; therefore, $\mathbf{w}^T\mathbf{x}$ in (2)–(4) is replaced with $\mathbf{w}^T\mathbf{x}+b$. For example, the logistic loss function becomes

$$\xi_{\log}(\mathbf{w},b;\mathbf{x},y) = \log\left(1+e^{-y(\mathbf{w}^T\mathbf{x}+b)}\right).$$

The optimization problem then involves variables $\mathbf{w}$ and b:

$$\min_{\mathbf{w},b} \quad \|\mathbf{w}\|_1 + C\sum_{i=1}^{l}\xi(\mathbf{w},b;\mathbf{x}_i,y_i). \tag{5}$$

Because b does not appear in the regularization term, most optimization methods used to solve (1) can solve (5) as well. In this paper, except whenever b is required, we mainly consider the formulation (1).

Many papers have proposed optimization methods for large-scale L1-regularized logistic regression (i.e., using $\xi_{\log}$ as the loss function). These studies did not consider L1- or L2-loss functions because logistic loss has a probability interpretation and is twice differentiable. These methods differ in various aspects such as the convergence speed, ease of implementation, and practical applicability. With so many available methods, it is important to conduct a comprehensive comparison. Schmidt et al. (2007, 2009) compared optimization methods for L1-regularized classifiers; however, they did not include some state-of-the-art solvers. Moreover, their comparison is based on the number of function evaluations instead of the actual running time. In this paper, we categorize and compare existing methods for logistic regression. We also extend some methods to solve L2-loss SVMs. We exclude L1 loss from the discussion because most methods for logistic regression or L2-loss SVMs assume the differentiability of the loss functions. Readers interested in using L1-loss SVMs can refer to Bradley and Mangasarian (1998), Zhu et al. (2004), Fung and Mangasarian (2004), Mangasarian (2006), and references therein.

1.1 Basic Properties of (1)

In this paper, we assume that the loss function $\xi(\mathbf{w};\mathbf{x}_i,y_i)$ is convex, differentiable, and nonnegative. The proof presented in Appendix A shows that (1) attains at least one global optimum. Unlike L2 regularization, which leads to a unique optimal solution, here, (1) may possess multiple optimal solutions. We use $\mathbf{w}^*$ to denote any optimal solution of (1). The convexity of $f(\mathbf{w})$ implies that all optimal solutions have the same function value, which is denoted as f^*. For more information about the set of optimal solutions, see, for example, Hale et al. (2008, Section 2).

From standard convex analysis, $\mathbf{w}^*$ is optimal for (1) if and only if $\mathbf{w}^*$ satisfies the following optimality conditions:

$$\begin{cases} \nabla_j L(\mathbf{w}^*) + 1 = 0 & \text{if } w_j^* > 0, \\ \nabla_j L(\mathbf{w}^*) - 1 = 0 & \text{if } w_j^* < 0, \\ -1 \le \nabla_j L(\mathbf{w}^*) \le 1 & \text{if } w_j^* = 0, \end{cases} \tag{6}$$

where $L(\mathbf{w})$ is the loss term in (1):

$$L(\mathbf{w}) \equiv C \sum_{i=1}^{l} \xi(\mathbf{w};\mathbf{x}_i,y_i). \tag{7}$$

1.2 Organization and Notation

The remainder of this paper is organized as follows. In Section 2, we briefly survey existing approaches. Section 3 lists state-of-the-art L1-regularized logistic regression packages compared in this study. Sections 4–6 discuss these packages in detail; two of these (Sections 4.1.2 and 5.1) are our proposed implementations. In Section 7, we extend several methods to train L2-loss SVMs. Section 8 describes the extensive experiments that were carried out. Comparison results indicate that decomposition methods (Section 4) are very suitable for large-scale document data. Finally, the discussions and conclusions are presented in Section 9. A supplementary file including additional details and descriptions of experiments is available at `http://www.csie.ntu.edu.tw/~cjlin/papers/l1/supplement.pdf`

In this study, we use consistent notation to describe several state-of-the-art methods that are considered. The following table lists some frequently used symbols:

l:	number of instances
n:	number of features
i:	index for a data instance
j:	index for a data feature
k:	iteration index for an optimization algorithm

We may represent training instances $(\mathbf{x}_i,y_i)$, $i=1,\ldots,l$ in the following form:

$$X = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_l^T \end{bmatrix} \in R^{l\times n} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_l \end{bmatrix} \in \{-1,+1\}^l.$$

For any vector $\mathbf{v}$, we consider the following two representations for sub-vectors:

$$\mathbf{v}_{t:s} \equiv [v_t,\ldots,v_s]^T \quad \text{and} \quad \mathbf{v}_I \equiv [v_{i_1},\ldots,v_{i_{|I|}}]^T,$$

where $I = \{i_1, \ldots, i_{|I|}\}$ is an index set. Similarly,

$$X_{I,:} \equiv \begin{bmatrix} \mathbf{x}_{i_1}^T \\ \vdots \\ \mathbf{x}_{i_{|I|}}^T \end{bmatrix}. \tag{8}$$

The function $\tau(s)$ gives the first derivative of the logistic loss function $\log(1+e^s)$:

$$\tau(s) \equiv \frac{1}{1+e^{-s}}. \tag{9}$$

An indicator vector for the jth component is

$$\mathbf{e}_j \equiv [\underbrace{0, \ldots, 0}_{j-1}, 1, 0, \ldots, 0]^T. \tag{10}$$

We use $\|\cdot\|$ or $\|\cdot\|_2$ to denote the 2-norm and $\|\cdot\|_1$ to denote the 1-norm.

2. A Survey of Existing Methods

In this section, we survey existing methods for L1-regularized problems. In Sections 2.1–2.3, we focus on logistic regression and L2-loss SVMs for data classification. Section 2.5 briefly discusses works on regression problems using the least-square loss.

2.1 Decomposition Methods

Decomposition methods are a classical optimization approach. Because it is difficult to update all variables simultaneously, at each iteration, we can choose a subset of variables as the working set and fix all others. The resulting sub-problem contains fewer variables and is easier to solve. The various decomposition methods that have been applied to solve L1-regularized problems are categorized into two types according to the selection of the working set.

2.1.1 CYCLIC COORDINATE DESCENT METHODS

A simple coordinate descent method cyclically chooses one variable at a time and solves the following one-variable sub-problem:

$$\min_z \quad g_j(z) \equiv f(\mathbf{w} + z\mathbf{e}_j) - f(\mathbf{w}), \tag{11}$$

where $\mathbf{e}_j$ is defined in (10). This function $g_j(z)$ has only one non-differentiable point at $z = -w_j$. In optimization, the cyclic method for choosing working variables is often called the Gauss-Seidel rule (Tseng and Yun, 2007).

Several works have applied coordinate descent methods to solve (1) with logistic loss. Here, a difficulty arises in that sub-problem (11) does not have a closed-form solution. Goodman (2004) assumed nonnegative feature values (i.e., $x_{ij} \geq 0,\ \forall i, j$) and then approximated $g_j(z)$ by a function $A_j(z)$ at each iteration. $A_j(z)$ satisfies $A_j(z) \geq g_j(z), \forall z$, and $A_j(0) = g_j(0) = 0$; therefore, minimizing $A_j(z)$ may reduce the function value. Moreover, there is a closed-form solution for minimizing $A_j(z)$. Goodman (2004) actually studied a problem with additional constraints $w_j \geq 0$; however, his

approach can be extended to solve the original L1-regularized logistic regression by taking the sign of w_j into consideration.

Genkin et al. (2007) implemented a cyclic coordinate descent method called BBR to solve L1-regularized logistic regression. BBR approximately minimizes sub-problem (11) in a trust region and applies one-dimensional Newton steps. Balakrishnan and Madigan (2005) reported an extension of BBR for online settings.

In Section 4.1.2, we propose a coordinate descent method by extending Chang et al.'s (2008) approach for L2-regularized classifiers. Chang et al. (2008) approximately solved the sub-problem (11) by a one-dimensional Newton direction with line search. Experiments show that their approach outperforms a BBR variant for L2-regularized classification. Therefore, for L1 regularization, this approach might be faster than BBR. Hereafter, we refer to this efficient coordinate descent method as CDN (coordinate descent using one-dimensional Newton steps).

Tseng and Yun (2007) broadly discussed decomposition methods for L1-regularized problems. One of their approaches is a cyclic coordinate descent method. They considered a general cyclic setting so that several working variables are updated at each iteration. We show that CDN is a special case of their general algorithms.

If we randomly select the working variable, then the procedure becomes a stochastic coordinate descent method. Shalev-Shwartz and Tewari (2009, Algorithm 2) recently studied this issue. Duchi and Singer (2009) proposed a similar coordinate descent method for the maximum entropy model, which is a generalization of logistic regression.

2.1.2 VARIABLE SELECTION USING GRADIENT INFORMATION

Instead of cyclically updating one variable, we can choose working variables based on the gradient information.[1] This method for selecting variables is often referred to as the Gauss-Southwell rule (Tseng and Yun, 2007). Because of the use of gradient information, the number of iterations is fewer than those in cyclic coordinate descent methods. However, the cost per iteration is higher. Shevade and Keerthi (2003) proposed an early decomposition method with the Gauss-Southwell rule to solve (1). In their method, one variable is chosen at a time and one-dimensional Newton steps are applied; therefore, their method differs from the cyclic coordinate descent methods described in Section 2.1.1 mainly in terms of finding the working variables. Hsieh et al. (2008) showed that for L2-regularized linear classification, maintaining the gradient for selecting only one variable at a time is not cost-effective. Thus, for decomposition methods using the gradient information, a larger working set should be used at each iteration.

In the framework of decomposition methods proposed by Tseng and Yun (2007), one type of method selects a set of working variables based on the gradient information. The working set can be large, and therefore, they approximately solved the sub-problem. For the same method, Yun and Toh (2009) enhanced the theoretical results and carried out experiments with document classification data. We refer to their method as CGD-GS because the method described in Tseng and Yun (2007) is called "coordinate gradient descent" and a Gauss-Southwell rule is used.

2.1.3 ACTIVE SET METHODS

Active set methods are a popular optimization approach for linear-constrained problems. For problem (1), an active method becomes a special decomposition method because at each iteration, a

1. Because $f(\mathbf{w})$ is not differentiable, $\nabla_j L(\mathbf{w}) \pm 1$ is used according to the sign of w_j.

sub-problem over a working set of variables is solved. The main difference from decomposition methods described earlier is that the working set contains all non-zero variables. Therefore, an active set method iteratively predicts what a correct split of zero and non-zero elements in $\mathbf{w}$ is. If the split is correct, then solving the sub-problem gives the optimal values of non-zero elements.

Perkins et al. (2003) proposed an active set method for (1) with logistic loss. This implementation uses gradient information to predict $\mathbf{w}$'s zero and non-zero elements.

2.2 Methods by Solving Constrained Optimization Problems

This type of method transforms (1) to equivalent constrained optimization problems. We further classify them into two groups.

2.2.1 Optimization Problems with Smooth Constraints

We can replace $\mathbf{w}$ in (1) with $\mathbf{w}^+ - \mathbf{w}^-$, where $\mathbf{w}^+$ and $\mathbf{w}^-$ are both nonnegative vectors. Then, problem (1) becomes equivalent to the following bound-constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{w}^+,\mathbf{w}^-} \quad & \sum_{j=1}^{n} w_j^+ + \sum_{j=1}^{n} w_j^- + C\sum_{i=1}^{l} \xi(\mathbf{w}^+ - \mathbf{w}^-; \mathbf{x}_i, y_i) \\ \text{subject to} \quad & w_j^+ \geq 0,\ w_j^- \geq 0, \quad j = 1,\ldots,n. \end{aligned} \tag{12}$$

The objective function and constraints of (12) are smooth, and therefore, the problem can be solved by standard bound-constrained optimization techniques. Schmidt et al. (2009) proposed ProjectionL1 to solve (12). This is an extension of the "two-metric projection" method (Gafni and Bertsekas, 1984). Limited-memory quasi Newton implementations, for example, LBFGS-B by Byrd et al. (1995) and BLMVM by Benson and Moré (2001), require function/gradient evaluations and use a limited amount of memory to approximate the Hessian matrix. Kazama and Tsujii (2003) presented an example of using BLMVM for (12). Lin and Moré's (1999) trust region Newton method (TRON) can also be applied to solve (12). In addition to function/gradient evaluations, TRON needs to calculate Hessian-vector products for faster convergence. Lin et al. (2008) applied TRON to solve L2-regularized logistic regression and showed that TRON outperforms LBFGS for document data. No previous work has applied TRON to solve L1-regularized problems, and therefore, we describe this in Section 5.1.

Koh et al. (2007) proposed an interior point method to solve L1-regularized logistic regression. They transformed (1) to the following constrained problem:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{u}} \quad & \sum_{j=1}^{n} u_j + C\sum_{i=1}^{l} \xi(\mathbf{w}; \mathbf{x}_i, y_i) \\ \text{subject to} \quad & -u_j \leq w_j \leq u_j, \quad j = 1,\ldots,n. \end{aligned} \tag{13}$$

Equation (13) can be made equivalent to (12) by setting

$$w_j^+ = \frac{u_j + w_j}{2} \text{ and } w_j^- = \frac{u_j - w_j}{2}.$$

To ensure that $\mathbf{w}$ and $\mathbf{u}$ are in the interior of the feasible region set, Koh et al. (2007) added a log barrier to the objective function of (13) and applied a Newton method.

2.2.2 OPTIMIZATION PROBLEMS WITH NON-SMOOTH CONSTRAINTS

It is well-known that for any choice of C in (1), there exists a corresponding K such that (1) is equivalent to the following problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \sum_{i=1}^{l} \xi(\mathbf{w}; \mathbf{x}_i, y_i) \\ \text{subject to} \quad & \|\mathbf{w}\|_1 \leq K. \end{aligned} \tag{14}$$

See the explanation in, for example, Donoho and Tsaig (2008, Section 1.2). Notice that in (14), the constraint is not smooth at $\{\mathbf{w} \mid w_j = 0 \text{ for some } j\}$. However, (14) contains fewer variables and constraints than (12). Lee et al. (2006) applied the LARS algorithm described in Efron et al. (2004) to find a Newton direction at each step and then used a backtracking line search to minimize the objective value. Kivinen and Warmuth's (1997) concept of exponentiated gradient (EG) can solve (14) with additional constraints $w_j \geq 0,\ \forall j$. In a manner similar to the technique for constructing (12), we can remove the nonnegative constraints by replacing $\mathbf{w}$ with $\mathbf{w}^+ - \mathbf{w}^-$. If k is the iteration index, EG updates $\mathbf{w}$ by the following rule:

$$w_j^{k+1} = \frac{w_j^k \exp\left(-\eta_k \nabla_j \left(\sum_{i=1}^{l} \xi(\mathbf{w}^k; \mathbf{x}_i, y_i)\right)\right)}{Z_k},$$

where Z_k is a normalization term for maintaining $\|\mathbf{w}^k\|_1 = K,\ \forall k$ and η_k is a user-defined learning rate. Duchi et al. (2008) applied a gradient projection method to solve (14). The update rule is

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}} \left\{ \left\| \left(\mathbf{w}^k - \eta_k \nabla\left(\sum_{i=1}^{l} \xi(\mathbf{w}^k; \mathbf{x}_i, y_i)\right)\right) - \mathbf{w} \right\| \;\middle|\; \|\mathbf{w}\|_1 \leq K \right\}. \tag{15}$$

They developed a fast algorithm to project a solution to the closest point satisfying the constraint. They also considered replacing the gradient in (15) with a stochastic sub-gradient. In a manner similar to (15), Liu et al. (2009) proposed a gradient projection method called Lassplore and carefully addressed the selection of the learning rate η_k. However, they evaluated their method only on data with no more than 2,000 instances.[2]

Kim and Kim (2004) discussed a coordinate descent method to solve (14). They used the gradient information to select an element w_j for update. However, because of the constraint $\|\mathbf{w}\|_1 \leq K$, the whole vector $\mathbf{w}$ is normalized at each step. Thus, the setting is very different from the unconstrained situations described in Section 2.1.1. Kim et al. (2008) made further improvements to realize faster convergence.

Active set methods have also been applied to solve (14). However, in contrast to the active set methods described in Section 2.1.3, here, the sub-problem at each iteration is a constrained optimization problem. Roth (2004) studied general loss functions including logistic loss.

2.3 Other Methods for L1-regularized Logistic Regression/L2-loss SVMs

We briefly review other methods in this section.

2. Although Table 1 in Liu et al. (2009) shows larger numbers, in Section 6 of the same paper, they stated that "we use a total of 2,000 samples in the following experiments."

2.3.1 EXPECTATION MAXIMIZATION

Many studies have considered Expectation Maximization (EM) frameworks to solve (1) with logistic loss (e.g., Figueiredo, 2003; Krishnapuram et al., 2004, 2005). These works find an upper-bound function $\hat{f}(\mathbf{w}) \geq f(\mathbf{w}), \forall \mathbf{w}$, and perform Newton steps to minimize $\hat{f}(\mathbf{w})$. However, as pointed out by Schmidt et al. (2009, Section 3.2), the upper-bound function $\hat{f}(\mathbf{w})$ may not be well-defined at some $w_i = 0$ and hence certain difficulties must be addressed.

2.3.2 STOCHASTIC GRADIENT DESCENT

Stochastic gradient descent methods have been successfully applied to solve (1). At each iteration, the solution is updated using a randomly selected instance. These types of methods are known to efficiently generate a reasonable model, although they suffer from slow local convergence. Under an online setting, Langford et al. (2009) solved L1-regularized problems by a stochastic gradient descent method. Shalev-Shwartz and Tewari (2009) combined Langford et al.'s (2009) method with other techniques to obtain a new stochastic gradient descent method for (1).

2.3.3 QUASI NEWTON METHODS

Andrew and Gao (2007) proposed an Orthant-Wise Limited-memory quasi Newton (OWL-QN) method. This method is extended from LBFGS (Liu and Nocedal, 1989), a limited memory quasi Newton approach for unconstrained smooth optimization. At each iteration, this method finds a sub-space without considering some dimensions with $w_j = 0$ and obtains a search direction similar to LBFGS. A constrained line search on the same sub-space is then conducted and the property $w_j^{k+1} w_j^k \geq 0$ is maintained. Yu et al. (2010) proposed a quasi Newton approach to solve non-smooth convex optimization problems. Their method can be used to improve the line search procedure in OWL-QN.

2.3.4 HYBRID METHODS

Shi et al. (2010) proposed a hybrid algorithm for (1) with logistic loss. They used a fixed-point method to identify the set $\{j \mid w_j^* = 0\}$, where $\mathbf{w}^*$ is an optimal solution, and then applied an interior point method to achieve fast local convergence.

2.3.5 QUADRATIC APPROXIMATION FOLLOWED BY COORDINATE DESCENT

Krishnapuram et al. (2005) and Friedman et al. (2010) replaced the loss term with a second-order approximation at the beginning of each iteration and then applied a cyclic coordinate descent method to minimize the quadratic function. We will show that an implementation in Friedman et al. (2010) is efficient.

2.3.6 CUTTING PLANE METHODS

Teo et al. (2010) implemented a bundle (cutting plane) method BMRM to handle non-smooth loss functions. It includes an extension for L1 regularization.

2.3.7 APPROXIMATING L1 REGULARIZATION BY L2 REGULARIZATION

Kujala et al. (2009) proposed the approximation of L1-regularized SVMs by iteratively reweighting training data and solving L2-regularized SVMs. That is, using the current w_j, they adjusted the jth feature of the training data and then trained an L2-regularized SVM in the next step.

2.3.8 SOLUTION PATH

Several works have attempted to find the "solution path" of (1). The optimal solution of (1) varies according to parameter C. It is occasionally useful to find all solutions as a function of C; see, for example, Rosset (2005), Zhao and Yu (2007), Park and Hastie (2007), and Keerthi and Shevade (2007). We do not provide details of these works because this paper focuses on the case in which a single C is used.

2.4 Strengths and Weaknesses of Existing Methods

Although this paper will compare state-of-the-art software, we discuss some known strengths and weaknesses of existing methods.

2.4.1 CONVERGENCE SPEED

Optimization methods using higher-order information (e.g., quasi Newton or Newton methods) usually enjoy fast local convergence. However, they involve an expensive iteration. For example, Newton methods such as TRON or IPM need to solve a large linear system related to the Hessian matrix. In contrast, methods using or partially using the gradient information (e.g., stochastic gradient descent) have slow local convergence although they can more quickly decrease the function value in the early stage.

2.4.2 IMPLEMENTATION EFFORTS

Methods using higher-order information are usually more complicated. Newton methods need to include a solver for linear systems. In contrast, methods such as coordinate descent or stochastic gradient descent methods are easy to implement. They involve only vector operations. Other methods such as expectation maximization are intermediate in this respect.

2.4.3 HANDLING LARGE-SCALE DATA

In some methods, the Newton step requires solving a linear system of n variables. Inverting an $n \times n$ matrix is difficult for large n. Thus, one should not use direct methods (e.g., Gaussian elimination) to solve the linear system. Instead, TRON and IPM employ conjugate gradient methods and Friedman et al. (2007) use coordinate descent methods. We observe that for existing EM implementations, many consider direct inversions, and therefore, they cannot handle large-scale data.

2.4.4 FEATURE CORRELATION

Methods that work on a block of variables at a time (e.g., decomposition methods) may be more efficient if features are almost independent; however, they may be less efficient if features are highly correlated.

2.4.5 DATA TYPE

No method is the most suitable for all data types. A method that is efficient for one application may be slow for another. This paper is focused on document classification, and a viable method must be able to easily handle large and sparse features.

2.5 Least-square Regression for Signal Processing and Image Applications

Recently, L1-regularized problems have attracted considerable attention for signal processing and image applications. However, they differ from (1) in several aspects. First, $y_i \in R$, and therefore, a regression instead of a classification problem is considered. Second, the least-square loss function is used:

$$\xi(\mathbf{w};\mathbf{x}_i,y_i) = (y_i - \mathbf{w}^T\mathbf{x}_i)^2. \tag{16}$$

Third, in many situations, $\mathbf{x}_i$ are not directly available. Instead, the product between the data matrix X and a vector can be efficiently obtained through certain operators. We briefly review some of the many optimization methods for least-square regression. If we use formulation (14) with non-smooth constraints, the problem reduces to LASSO proposed by Tibshirani (1996) and some early optimization studies include, for example, Fu (1998) and Osborne et al. (2000a,b). Fu (1998) applied a cyclic coordinate descent method. For least-square loss, the minimization of the one-variable sub-problem (11) has a closed-form solution. Sardy et al. (2000) also considered coordinate descent methods, although they allowed a block of variables at each iteration. Wu and Lange (2008) considered a coordinate descent method, but used the gradient information for selecting the working variable at each iteration. Osborne et al. (2000a) reported one of the earliest active set methods for L1-regularized problems. Roth's (2004) method for general losses (see Section 2.2.2) reduces to this method if the least-square loss is used. Friedman et al. (2007) extended Fu's coordinate descent method to find a solution path. Donoho and Tsaig (2008) also obtained a solution path. Their procedure requires solving a linear system of a matrix $X_{:,J}^T X_{:,J}$, where J is a subset of $\{1,\ldots,n\}$. Figueiredo et al. (2007) transformed the regression problem to a bound-constrained formula in (12) and applied a projected gradient method. Wright et al. (2009) proposed the iterative minimization of the sum of the L1 regularization term and a quadratic approximation of the loss term. In the quadratic approximation, they used a diagonal matrix instead of the Hessian of the loss term, and therefore, the minimization can be easily carried out. Hale et al. (2008) proposed a fixed point method to solve (1) with the least-square loss (16). Their update rule is generated from a fixed-point view; however, it is very similar to a gradient projection algorithm.

The dual of LASSO and the dual of (1) have been discussed in Osborne et al. (2000b) and Kim et al. (2007), respectively. However, thus far, there have been few optimization methods for the dual problem. Tomioka and Sugiyama (2009) proposed a dual augmented Lagrangian method for L1-regularized least-square regression that theoretically converges super-linearly.

Most optimization methods for classification discussed in the earlier sections can handle general smooth loss functions, and therefore, they can be applied to the regression problem. However, data for classification applications may be very different from those for regression applications. For example, in text classification, the numbers of instances and features are both large and data are very sparse. However, in certain signal processing applications, the number of instances may be much smaller than the number of features (i.e., $l \ll n$) and the data may be dense. Moreover, in classification, the parameter C is often selected by cross validation; however, in signal processing, the selection may be application dependent. Few optimization studies have investigated both types

of applications. The interior point method for logistic regression by solving (13) has been applied to the least-square regression problem (Kim et al., 2007). Duchi et al. (2008) compared their gradient projection implementation (see Section 2.2.2) with interior point methods using both logistic and least-square losses. In Section 2.1.2, we mentioned a decomposition method CGD-GS by Yun and Toh (2009). In the same paper, Yun and Toh have also investigated the performance of CGD-GS on regression problems.

In this paper, we focus on data classification, and therefore, our conclusions may not apply to regression applications. In particular, the efficient calculation between X and a vector in some signal processing applications may afford advantages to some optimization methods.

3. Methods and Software Included for Comparison

In the rest of this paper, we compare some state-of-the-art software BBR, SCD, CGD-GS, IPM, BMRM, OWL-QN, Lassplore and GLMNET. We further develop two efficient implementations. One is a coordinate descent method (CDN) and the other is a trust region Newton method (TRON). These packages are selected because of two reasons. First, they are publicly available. Second, they are able to handle large and sparse data sets. We categorize these packages into three groups:

- decomposition methods,
- methods by solving constrained optimization problems,
- other methods,

and describe their algorithms in Sections 4–6. The comparison results are described in Sections 8. Note that classification (logistic regression and L2-loss SVMs) is our focus, and therefore, software for regression problems using the least-square loss (16) is not considered.

4. Decomposition Methods

This section discusses three methods sequentially or randomly choosing variables for update, and one method using gradient information for the working set selection.

4.1 Cyclic Coordinate Descent Methods

From the current solution $\mathbf{w}^k$, a coordinate descent method updates one variable at a time to generate $\mathbf{w}^{k,j} \in R^n,\ j = 1,\ldots,n+1$, such that $\mathbf{w}^{k,1} = \mathbf{w}^k$, $\mathbf{w}^{k,n+1} = \mathbf{w}^{k+1}$, and

$$\mathbf{w}^{k,j} = \left[w_1^{k+1},\ldots,w_{j-1}^{k+1},w_j^k,\ldots,w_n^k\right]^T \text{ for } j = 2,\ldots,n.$$

To update $\mathbf{w}^{k,j}$ to $\mathbf{w}^{k,j+1}$, the following one-variable optimization problem is solved:

$$\min_z \quad g_j(z) = |w_j^{k,j} + z| - |w_j^{k,j}| + L_j(z;\mathbf{w}^{k,j}) - L_j(0;\mathbf{w}^{k,j}), \tag{17}$$

where

$$L_j(z;\mathbf{w}^{k,j}) \equiv L(\mathbf{w}^{k,j} + z\mathbf{e}_j).$$

Algorithm 1 A framework of coordinate descent methods

1. Given $\mathbf{w}^1$.
2. For $k = 1, 2, 3, \ldots$ (outer iterations)
   (a) $\mathbf{w}^{k,1} = \mathbf{w}^k$.
   (b) For $j = 1, 2, \ldots, n$ (inner iterations)
      - Find d by solving the sub-problem (17) exactly or approximately.
      - $\mathbf{w}^{k,j+1} = \mathbf{w}^{k,j} + d\mathbf{e}_j$.
   (c) $\mathbf{w}^{k+1} = \mathbf{w}^{k,n+1}$.

For simplicity, hereafter, we use $L_j(z)$ in most places. If the solution of (17) is d, then we update the jth element by

$$w_j^{k,j+1} = w_j^{k,j} + d.$$

Typically, a coordinate descent method sequentially goes through all variables and then repeats the same process; see the framework in Algorithm 1. We refer to the process of updating every n elements (i.e., from $\mathbf{w}^k$ to $\mathbf{w}^{k+1}$) as an outer iteration and the step of updating one element (i.e., from $\mathbf{w}^{k,j}$ to $\mathbf{w}^{k,j+1}$) as an inner iteration. Practically, we only approximately solve (17), where several approaches are discussed below.

Using the same way to obtain the optimality condition in (6), for the one-variable function $g_j(z)$, we have that $z = 0$ is optimal for (17) if and only if

$$\begin{cases} L_j'(0) + 1 = 0 & \text{if } w_j^{k,j} > 0, \\ L_j'(0) - 1 = 0 & \text{if } w_j^{k,j} < 0, \\ -1 \le L_j'(0) \le 1 & \text{if } w_j^{k,j} = 0. \end{cases} \tag{18}$$

The optimality condition at $z = 0$ shows if modifying $w_j^{k,j}$ may decrease the function value or not.

4.1.1 BBR

Genkin et al. (2007) propose a coordinate descent method BBR for (1) and (5) with logistic loss. This method is extended from Zhang and Oles (2001) for L2-regularized logistic regression. At each inner iteration, BBR approximately solves the sub-problem (17) by a trust region Newton method. With the trust region Δ_j, it requires the step z to satisfy

$$|z| \le \Delta_j \quad \text{and} \quad w_j^{k,j} + z \begin{cases} \ge 0 & \text{if } w_j^{k,j} > 0, \\ \le 0 & \text{if } w_j^{k,j} < 0. \end{cases} \tag{19}$$

The first constraint confines the step size to be in the trust region and Δ_j is updated at the end of each inner iteration. Due to the non-differentiable point at $z = -w_j^{k,j}$, the second constraint ensures that the function is differentiable in the search space.

To approximately solve (17), BBR minimizes a quadratic function upper-bounding the function $g_j(z)$ in the trust region. Though $g_j(z)$ is not differentiable, by considering both cases of $w_j^{k,j} > 0$ and $w_j^{k,j} < 0$, we obtain the following form:

$$g_j(z) = g_j(0) + g_j'(0)z + \frac{1}{2} g_j''(\eta z) z^2, \tag{20}$$

where $0 < \eta < 1$,

$$g'_j(0) \equiv \begin{cases} L'_j(0)+1 & \text{if } w_j^{k,j} > 0, \\ L'_j(0)-1 & \text{if } w_j^{k,j} < 0, \end{cases} \text{ and } g''_j(\eta z) \equiv L''_j(\eta z). \tag{21}$$

Notice that when $w_j^{k,j} = 0$, $g_j(z)$ is non-differentiable at $z = 0$ and $g'_j(0)$ is not well-defined. We will discuss this situation later. BBR finds an upper bound U_j of $g''_j(z)$ such that

$$U_j \geq g''_j(z),\ \forall |z| \leq \Delta_j.$$

Then $\hat{g}_j(z)$ is an upper-bound function of $g_j(z)$:

$$\hat{g}_j(z) \equiv g_j(0) + g'_j(0)z + \frac{1}{2}U_j z^2.$$

Any step z satisfying $\hat{g}_j(z) < \hat{g}_j(0)$ leads to

$$g_j(z) - g_j(0) = g_j(z) - \hat{g}_j(0) \leq \hat{g}_j(z) - \hat{g}_j(0) < 0,$$

so the function value is decreased. If logistic loss (2) is used, BBR suggests setting U_j as

$$U_j \equiv C \sum_{i=1}^{l} x_{ij}^2 F\left(y_i (\mathbf{w}^{k,j})^T \mathbf{x}_i, \Delta_j |x_{ij}|\right), \tag{22}$$

where

$$F(r,\delta) = \begin{cases} 0.25 & \text{if } |r| \leq \delta, \\ \frac{1}{2+e^{(|r|-\delta)}+e^{(\delta-|r|)}} & \text{otherwise.} \end{cases}$$

If $w_j^{k,j} \neq 0$, BBR minimizes $\hat{g}_j(z)$ under the constraints (19) to obtain

$$d = \min\left(\max\left(P(-\frac{g'_j(0)}{U_j}, w_j^{k,j}), -\Delta_j\right), \Delta_j\right), \tag{23}$$

where

$$P(z,w) \equiv \begin{cases} z & \text{if } \operatorname{sgn}(w+z) = \operatorname{sgn}(w), \\ -w & \text{otherwise.} \end{cases}$$

Now consider the case of $w_j^{k,j} = 0$, where $g'_j(0)$ is not well-defined at this point. If $L'_j(0)+1 < 0$, by defining $g'_j(0) \equiv L'_j(0)+1$, any $0 < z \leq -g'_j(0)/U_j$ gives a smaller $\hat{g}_j(z)$ than $\hat{g}_j(0)$. We thus obtain the new point by mapping $-g'_j(0)/U_j$ back to the trust region. The situation for $L'_j(0)-1 > 0$ is similar. We do not need to handle the situation $-1 \leq L'_j(0) \leq 1$ as (18) and $w_j^{k,j} = 0$ indicate that $z = 0$ is the minimum of $g_j(z)$.

The procedure of BBR to approximately solve (17) is described in Algorithm 2. The major cost is on calculating $g'_j(0)$ and U_j. For logistic loss, $L'_j(0)$ needed for calculating $g'_j(0)$ is

$$L'_j(0) = C \sum_{i=1}^{l} y_i x_{ij} \left(\tau(y_i (\mathbf{w}^{k,j})^T \mathbf{x}_i) - 1\right), \tag{24}$$

Algorithm 2 BBR: Approximately solving the sub-problem by a trust region method

1. Given $\mathbf{w}^{k,j}$ and Δ_j.
2. Calculate U_j by (22).
3. Find a step d by (23).
4. Update Δ_j by $\Delta_j \leftarrow \max(2|d|, \Delta_j/2)$.

where $\tau(\cdot)$ is defined in (9). From (22) and (24), the most expensive operation is on obtaining $\mathbf{w}^T\mathbf{x}_i$, $\forall i$. A common trick for saving the running time is to store $\mathbf{w}^T\mathbf{x}_i$, $\forall i$ and update them according to

$$\mathbf{w}^T\mathbf{x}_i \leftarrow \mathbf{w}^T\mathbf{x}_i + d \cdot x_{ij}. \tag{25}$$

If $\mathbf{w}^T\mathbf{x}_i$, $\forall i$ are available, both (22) and (24) need $O(l)$ operations. Because maintaining $\mathbf{w}^T\mathbf{x}_i$ via (25) also takes $O(l)$, the cost per inner iteration is $O(l)$.

Unfortunately, there is no convergence proof yet for the method BBR.

4.1.2 COORDINATE DESCENT METHOD USING ONE-DIMENSIONAL NEWTON DIRECTIONS (CDN)

BBR replaces the second derivative term in (20) with an upper bound U_j. If we keep using $g_j''(0)$ and obtain the one-dimensional Newton direction at $z = 0$, the local convergence may be faster. This issue has been studied in L2-regularized logistic regression/SVMs, where BBR reduces to the approach by Zhang and Oles (2001), and Chang et al. (2008) showed that a coordinate descent method using one-dimensional Newton directions is faster. Here, we extend Chang et al.'s method for L1-regularized problems. The new approach, referred to as CDN, is expected to be faster than BBR following a similar reason.

A Newton direction is obtained from minimizing the second-order approximation, but $g_j(z)$ is not differentiable due to the L1-regularization term. Thus, we consider only the second-order approximation of the loss term $L_j(z)$ and solve

$$\min_z \quad |w_j^{k,j} + z| - |w_j^{k,j}| + L_j'(0)z + \frac{1}{2}L_j''(0)z^2. \tag{26}$$

This problem can be reduced to a form commonly referred to as "soft-thresholding" in signal processing. We show in Appendix B that (26) has the following closed-form solution:

$$d = \begin{cases} -\frac{L_j'(0)+1}{L_j''(0)} & \text{if } L_j'(0)+1 \le L_j''(0)w_j^{k,j}, \\ -\frac{L_j'(0)-1}{L_j''(0)} & \text{if } L_j'(0)-1 \ge L_j''(0)w_j^{k,j}, \\ -w_j^{k,j} & \text{otherwise.} \end{cases} \tag{27}$$

Because (26) is only a quadratic approximation of $f(\mathbf{w}^{k,j} + z\mathbf{e}_j) - f(\mathbf{w}^{k,j})$, the direction d may not ensure the decrease of the function value. For the convergence, Chang et al. (2008) consider a line search procedure to find $\lambda \in (0,1)$ such that the step λd satisfies the sufficient decrease condition:

$$f(\mathbf{w}^{k,j} + \lambda d\mathbf{e}_j) - f(\mathbf{w}^{k,j}) = g_j(\lambda d) - g_j(0) \le -\sigma(\lambda d)^2,$$

where σ is any constant in $(0,1)$. However, as pointed out by Tseng and Yun (2007), this condition may not be suitable here due to the non-smooth regularization term $\|\mathbf{w}\|_1$. We follow Tseng and

Algorithm 3 CDN: Approximately solving the sub-problem by Newton directions with line search

1. Given $\mathbf{w}^{k,j}$. Choose $\beta \in (0,1)$.
2. Calculate the Newton direction d by (27).
3. Compute $\lambda = \max\{1, \beta, \beta^2, \dots\}$ such that λd satisfies (28).

Yun (2007) to use a modified condition:

$$g_j(\lambda d) - g_j(0) \le \sigma\lambda \left(L_j'(0)d + |w_j^{k,j} + d| - |w_j^{k,j}| \right). \tag{28}$$

To find λ, CDN adopts a backtrack line search to sequentially check $\lambda = 1,\ \beta,\ \beta^2,\ \dots$, where $\beta \in (0,1)$, until λd satisfies (28). A description of how CDN approximately solves the sub-problem (17) is in Algorithm 3.

In Appendix D, we explain that Algorithm 3 falls into a class of Gauss-Seidel coordinate descent methods in Tseng and Yun (2007). By showing that all required assumptions are satisfied, we can directly enjoy some nice theoretical properties. First, following Lemma 3.4(b) in Tseng and Yun (2007), the line search procedure stops in a finite number of steps. Second, for (1) with logistic loss, any limit point of $\{\mathbf{w}^k\}$ is an optimum.

We discuss the computational cost. To obtain the sub-problem (26), we must calculate $L_j'(0)$ and $L_j''(0)$. For logistic loss, $L_j'(0)$ is shown in (24) and

$$L_j''(0) = C \sum_{i=1}^{l} x_{ij}^2 \left(\tau(y_i(\mathbf{w}^{k,j})^T \mathbf{x}_i) \right) \left(1 - \tau(y_i(\mathbf{w}^{k,j})^T \mathbf{x}_i) \right). \tag{29}$$

Similar to the situation in BBR, calculating $\mathbf{w}^T\mathbf{x}_i,\ \forall i$ is the major cost here. We can apply the same trick in (25) to maintain $\mathbf{w}^T\mathbf{x}_i,\ \forall i$. In our implementation, we maintain $e^{\mathbf{w}^T\mathbf{x}_i}$ instead:

$$e^{\mathbf{w}^T\mathbf{x}_i} \leftarrow e^{\mathbf{w}^T\mathbf{x}_i} \cdot e^{\lambda d x_{ij}}. \tag{30}$$

The line search procedure needs to calculate $g_j(\lambda d)$. From (2), the main cost is still on obtaining $(\mathbf{w} + \lambda d\mathbf{e}_j)^T\mathbf{x}_i,\ \forall i$, so the trick in (30) is again useful. If $e^{\mathbf{w}^T\mathbf{x}_i},\ \forall i$ are available, from (24), (29) and (2), the cost per inner iteration is

$$(1 + \#\text{ line search steps}) \times O(l).$$

To reduce the cost for line search, Chang et al. (2008) obtain a function $\hat{g}_j(\cdot)$ satisfying $\hat{g}_j(\lambda d) > g_j(\lambda d)$ and check $\hat{g}_j(\lambda d) - g_j(0)$ in (28). Calculating $\hat{g}_j(\lambda d)$ is much easier than $g_j(\lambda d)$, so we may avoid calculating the last $g_j(\lambda d)$ in the line search procedure. In some iterations, $\lambda = 1$ already satisfies (28), and therefore, this trick makes the cost of the line search procedure negligible. We do not show details here; however, derivations can be found in Fan et al. (2008, Appendix G).

Next we describe two implementation techniques to improve the convergence speed. The first one is to use a random permutation of sub-problems. In Algorithm 1, we cyclically consider variables to form one-variable sub-problems. Chang et al. (2008) show that solving sub-problems in a random order may lead to faster convergence. That is, at the kth iteration, we randomly permute $\{1, 2, \dots, n\}$ to $\{\pi_k(1), \pi_k(2), \dots, \pi_k(n)\}$ and update the elements of $\mathbf{w}$ in the order of $\{w_{\pi_k(1)}, w_{\pi_k(2)}, \dots, w_{\pi_k(n)}\}$.

The second implementation technique is shrinking. Past works such as Hsieh et al. (2008), Joachims (1998), and Krishnapuram et al. (2005, Section 3.5) heuristically remove some variables to obtain a smaller optimization problem. If w_j has remained zero at many iterations, it is very possible that the optimal w_j^* is also zero. Therefore, we can remove such variables. Our shrinking implementation is related to that in the software LIBSVM (Chang and Lin, 2001). From the optimality condition (6),

$$-1 < \nabla_j L(\mathbf{w}^*) < 1 \quad \text{implies} \quad w_j^* = 0. \tag{31}$$

We prove in Appendix C the following theorem:

Theorem 1 *Assume $\{\mathbf{w}^k\}$ globally converges to $\mathbf{w}^*$. If $-1 < \nabla_j L(\mathbf{w}^*) < 1$, then there is K_j such that for all $k \geq K_j$,*

$$-1 < \nabla_j L(\mathbf{w}^{k,j}) < 1 \quad \textit{and} \quad w_j^{k,j} = 0.$$

Using this theorem, we design the following shrinking strategy. Before updating $w_j^{k,j}$ via approximately solving the sub-problem (17), if

$$w_j^{k,j} = 0 \quad \text{and} \quad -1 + M^{k-1} < \nabla_j L(\mathbf{w}^{k,j}) < 1 - M^{k-1}, \tag{32}$$

we conjecture that $w_j^{k,j}$ may remain zero and hence remove it for optimization. We choose

$$M^{k-1} \equiv \frac{\max_j v_j}{l},$$

where

$$v_j \equiv \begin{cases} |\nabla_j L(\mathbf{w}^{k-1,j}) + 1| & \text{if } w_j^{k-1,j} > 0, \\ |\nabla_j L(\mathbf{w}^{k-1,j}) - 1| & \text{if } w_j^{k-1,j} < 0, \\ \max\left(\nabla_j L(\mathbf{w}^{k-1,j}) - 1,\ -1 - \nabla_j L(\mathbf{w}^{k-1,j}),\ 0\right) & \text{if } w_j^{k-1,j} = 0, \end{cases}$$

From (6), $v_j, j = 1, \ldots, n$ measure the violation of the optimality condition at the $(k-1)$st iteration. The value M^{k-1} reflects how aggressive we remove variables. It is large in the beginning, but approaches zero in the end.

The shrinking implementation introduces little extra cost. When updating the jth component at the kth iteration, we calculate

$$\nabla_j L(\mathbf{w}^{k,j}) = L_j'(0; \mathbf{w}^{k,j})$$

for the direction d in (27), regardless of implementing shrinking or not. Thus, $\nabla_j L(\mathbf{w}^{k,j})$ needed for checking (32) is already available. Moreover, it can be used to calculate v_j and M^k, which are needed for the next iteration.

4.1.3 STOCHASTIC COORDINATE DESCENT METHOD (SCD)

Shalev-Shwartz and Tewari (2009) propose a stochastic coordinate descent method (SCD) to solve the bound-constrained problem in (12). At the kth iteration, SCD randomly chooses a working variable from $\{w_1^+, \ldots, w_n^+, w_1^-, \ldots, w_n^-\}$. The one-variable sub-problem is

$$\min_z \quad g_j(z) \equiv z + L_j(z; \mathbf{w}^{k,+} - \mathbf{w}^{k,-}) - L_j(0; \mathbf{w}^{k,+} - \mathbf{w}^{k,-}),$$

Algorithm 4 SCD for L1-regularized logistic regression

1. Given $(\mathbf{w}^+, \mathbf{w}^-)$ and $U_j > 0$.
2. While $(\mathbf{w}^+, \mathbf{w}^-)$ is not optimal for (12)
   (a) Select an element from $\{w_1^+, \ldots, w_n^+, w_1^-, \ldots, w_n^-\}$.
   (b) Update w_j^+ or w_j^- by (36)–(37).

subject to the non-negativity constraint

$$w_j^{k,+} + z \geq 0 \quad \text{or} \quad w_j^{k,-} + z \geq 0,$$

according to whether w_j^+ or w_j^- is the working variable. SCD considers a second-order approximation similar to BBR:

$$\hat{g}_j(z) = g_j(0) + g_j'(0)z + \frac{1}{2}U_j z^2, \tag{33}$$

where

$$g_j'(0) = \begin{cases} 1 + L_j'(0) & \text{for } w_j^+ \\ 1 - L_j'(0) & \text{for } w_j^- \end{cases} \quad \text{and} \quad U_j \geq g_j''(z),\ \forall z.$$

BBR considers U_j to be an upper bound of $g_j''(z)$ only in the trust region, while SCD finds a global upper bound of $g_j''(z)$. For logistic regression, following (9) and (29), we have $\tau(\cdot)\,(1-\tau(\cdot)) \leq 0.25$ and

$$U_j = 0.25C \sum_{i=1}^{l} x_{ij}^2 \geq g_j''(z),\ \forall z. \tag{34}$$

Shalev-Shwartz and Tewari (2009) assume $-1 \leq x_{ij} \leq 1,\ \forall i, j$, so a simpler upper bound is

$$U_j = 0.25Cl. \tag{35}$$

Using the direction obtained by minimizing (33) and taking the non-negativity into consideration, SCD updates $\mathbf{w}$ by the following way:

If w_j^+ is selected

$$w_j^+ \leftarrow w_j^+ + \max(-w_j^+,\ -\frac{1 + L_j'(0)}{U_j}) \tag{36}$$

Else

$$w_j^- \leftarrow w_j^- + \max(-w_j^-,\ -\frac{1 - L_j'(0)}{U_j}) \tag{37}$$

A description of SCD is in Algorithm 4.

4.2 CGD-GS: a Decomposition Method Using Gradients for Selecting Variables

Instead of updating one variable at a time, some decomposition methods choose a larger working set $J \subset N \equiv \{1, \ldots, n\}$. We discuss a Gauss-Southwell method for selecting J (Tseng and Yun, 2007; Yun and Toh, 2009). This method, referred to as CGD-GS, can handle any smooth loss function including $\xi_{\log}$.

Following the principle of decomposition methods, if $\mathbf{w}^k$ is the current solution and J is the set of working variables, one should solve the following sub-problem:

$$\begin{aligned} \min_{\mathbf{d}} \quad & L(\mathbf{w}^k+\mathbf{d})-L(\mathbf{w}^k)+\|\mathbf{w}^k+\mathbf{d}\|_1-\|\mathbf{w}^k\|_1 \\ \text{subject to} \quad & d_j=0,\ \forall j\notin J. \end{aligned}$$

Because it is still difficult to solve this sub-problem, CGD-GS considers a quadratic approximation of the loss term:

$$\begin{aligned} \min_{\mathbf{d}} \quad & q_k(\mathbf{d}) \equiv \nabla L(\mathbf{w}^k)^T\mathbf{d}+\frac{1}{2}\mathbf{d}^T H\mathbf{d}+\|\mathbf{w}^k+\mathbf{d}\|_1-\|\mathbf{w}^k\|_1 \\ \text{subject to} \quad & d_j=0,\ \forall j\notin J, \end{aligned} \tag{38}$$

where H is either $\nabla^2 L(\mathbf{w}^k)$ or its approximation. To ensure the convergence, CGD-GS conducts a backtrack line search to find λ such that $\lambda\mathbf{d}$ satisfies

$$f(\mathbf{w}^k+\lambda\mathbf{d})-f(\mathbf{w}^k)\le\sigma\lambda\left(\nabla L(\mathbf{w}^k)^T\mathbf{d}+\gamma\mathbf{d}^T H\mathbf{d}+\|\mathbf{w}^k+\mathbf{d}\|_1-\|\mathbf{w}^k\|_1\right), \tag{39}$$

where $0<\sigma<1$ and $0\le\gamma<1$. This condition is the same as (28) if $\gamma=0$ and J contains only one element. Tseng and Yun (2007) used (39) for both the cyclic selection (Gauss-Seidel) or the selection using gradient information (Gauss-Southwell).

For selecting J using gradients, Tseng and Yun (2007) proposed two possible ways. The first one, referred to as the Gauss-Southwell-r rule, requires J to satisfy

$$\|\mathbf{d}(J)\|_\infty\ge v\|\mathbf{d}(N)\|_\infty, \tag{40}$$

where $v\in(0,1)$ is a constant and $\mathbf{d}(J)$ and $\mathbf{d}(N)$ are the solution of (38) by considering J and N as the working set, respectively. This condition connects the directions of solving sub-problems using a subset and a full set. The other condition for selecting J is

$$q_k(\mathbf{d}(J))\le v\cdot q_k(\mathbf{d}(N)), \tag{41}$$

where $v\in(0,1)$ and $q_k(\mathbf{d})$ is defined in (38). This condition is referred to as the Gauss-Southwell-q rule. Algorithm 5 summarizes the procedure.

The remaining issues are how to solve the sub-problem (38) and how to obtain J satisfying (40) or (41). The CGD-GS implementation considers a diagonal matrix with positive entries as H. For example, $H_{jj}=\max(\nabla^2_{jj}L(\mathbf{w}^k),\varepsilon)$, where ε is a small positive value. Then, the sub-problem becomes $|J|$ separable one-variable problems like (26). Each one-variable problem has a simple closed-form solution. Further, it is easy to find indices satisfying (40) or (41). For example, the rule (40) becomes to find the larger elements of $\mathbf{d}(N)$. Tseng and Yun (2007) proved that any limit point of $\{\mathbf{w}^k\}$ is an optimum of (1).

We discuss the computational cost for logistic regression. If H is diagonal, then solving (38) takes only $O(|J|)$ operations. Constructing (38) is more expensive because we need to calculate

$$\nabla L(\mathbf{w})=C\sum_{i=1}^{l}\big(\tau(y_i\mathbf{w}^T\mathbf{x}_i)-1\big)y_i\mathbf{x}_i. \tag{42}$$

Yun and Toh (2009) apply a trick similar to (30) and maintain $e^{\mathbf{w}^T\mathbf{x}_i}$, $\forall i$, but the gradient calculation still needs $O(ln)$. This high cost may not pay off, and therefore, Hsieh et al. (2008) favor decomposition methods without maintaining the gradient. Finding the working set J is another potentially expensive step, but it is cheap for a diagonal H.

Algorithm 5 CGD-GS for L1-regularized logistic regression

1. Given $\mathbf{w}^1$. Choose (40) or (41) as the strategy for selecting working sets. Given $0 < \beta, \sigma < 1$ and $0 \le \gamma < 1$.
2. For $k = 1, 2, 3, \ldots$
   - Choose an H and a working set J.
   - Get $\mathbf{d}^k$ by solving the sub-problem (38).
   - Compute $\lambda = \max\{1, \beta, \beta^2, \ldots\}$ such that $\lambda \mathbf{d}^k$ satisfies (39).
   - $\mathbf{w}^{k+1} = \mathbf{w}^k + \lambda \mathbf{d}^k$.

5. Methods by Solving Constrained Optimization Problems

Section 2.2 lists several methods for L1-regularized classification by solving the bound-constrained problems (12), (13), and (14). In this section, we discuss TRON, IPM, and Lassplore in detail.

5.1 A Trust Region Newton Method (TRON)

We apply the trust region Newton method in Lin and Moré (1999) to solve (12). A previous study of this method for L1-regularized logistic regression is by Lee (2008). For convenience, in this section, we slightly abuse the notation by redefining

$$\mathbf{w} \equiv \begin{bmatrix} \mathbf{w}^+ \\ \mathbf{w}^- \end{bmatrix} \in R^{2n}. \tag{43}$$

Then, problem (12) can be written in a general form of bounded-constrained problems:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \bar{f}(\mathbf{w}) \\ \text{subject to} \quad & \mathbf{w} \in \Omega \equiv \{\mathbf{w} \mid l_j \le w_j \le u_j, \forall j\}, \end{aligned}$$

where $\bar{f}(\mathbf{w})$ denotes the objective function of (12), and $\mathbf{l}$ and $\mathbf{u}$ are lower and upper bounds, respectively.

At the kth iteration of the trust region Newton method, we have an iterate $\mathbf{w}^k$, a size Δ_k of the trust region, and a quadratic model

$$q_k(\mathbf{d}) \equiv \frac{1}{2}\mathbf{d}^T \nabla^2 \bar{f}(\mathbf{w}^k)\mathbf{d} + \nabla \bar{f}(\mathbf{w}^k)^T \mathbf{d}$$

to approximate the value $\bar{f}(\mathbf{w}^k + \mathbf{d}) - \bar{f}(\mathbf{w}^k)$. Next, we find a step $\mathbf{d}^k$ by approximately solving the following trust region problem

$$\begin{aligned} \min_{\mathbf{d}} \quad & q_k(\mathbf{d}) \\ \text{subject to} \quad & \|\mathbf{d}\| \le \Delta_k, \ \mathbf{w}^k + \mathbf{d} \in \Omega. \end{aligned} \tag{44}$$

We then update $\mathbf{w}^k$ and Δ_k by checking the ratio

$$\rho_k = \frac{\bar{f}(\mathbf{w}^k + \mathbf{d}^k) - \bar{f}(\mathbf{w}^k)}{q_k(\mathbf{d}^k)} \tag{45}$$

Algorithm 6 A trust region method for L1-regularized logistic regression

1. Given $\mathbf{w}^1$.
2. For $k = 1, 2, 3, \ldots$ (outer iterations)
   - Approximately solve (44) and obtain a direction $\mathbf{d}^k$; see Algorithm 7.
   - Compute ρ_k via (45).
   - Update $\mathbf{w}^k$ to $\mathbf{w}^{k+1}$ according to (46) and update Δ_k to Δ_{k+1}.

Algorithm 7 TRON: Finding a direction $\mathbf{d}^k$ by approximately solving (44)

1. Given $0 < \varepsilon < 1$. Find the Cauchy step $\mathbf{d}^{k,C}$ and the Cauchy point
$$\mathbf{w}^{k,1} = \mathbf{w}^k + \mathbf{d}^{k,C} \quad \text{and} \quad \mathbf{d}^{k,1} = \mathbf{d}^{k,C}.$$
2. For $t = 1, \ldots, 2n+1$ (inner iterations)
   - Find F_t and B_t (free/bounded sets) at $\mathbf{w}^{k,t}$ by (50).
   - If $F_t = \emptyset$, then stop and return $\mathbf{d}^k = \mathbf{w}^{k,t} - \mathbf{w}^k$.
   - Approximately solve
$$\begin{aligned} \min_{\mathbf{v}_{F_t}} \quad & q_k(\mathbf{d}^{k,t} + \mathbf{v}) \\ \text{subject to} \quad & \|\mathbf{d}^{k,t} + \mathbf{v}\| \le \Delta_k,\ \mathbf{v}_{B_t} = \mathbf{0}, \end{aligned}$$
     by Conjugate Gradient (CG) methods. Denote the solution as $\mathbf{v}^{k,t}$.
   - Projected line search on $\mathbf{w}^{k,t} + \lambda \mathbf{v}^{k,t}$ to obtain $\mathbf{w}^{k,t+1}$ and $\mathbf{d}^{k,t+1}$; see Equation (55). We ensure that $F_t \subset F_{t+1}$ and $|F_t| < |F_{t+1}|$.
   - If one of the following situations occurs:
$$\|\nabla q_k(\mathbf{d}^{k,t+1})_{F_t}\| \le \varepsilon \|\nabla \bar{f}(\mathbf{w}^k)_{F_t}\|,$$
     or CG abnormally stops (explained in text), then stop and return
$$\mathbf{d}^k = \mathbf{w}^{k,t+1} - \mathbf{w}^k.$$

of the actual reduction in the function to the predicted reduction in the quadratic model. The direction $\mathbf{d}^k$ is accepted if ρ_k is large enough:

$$\mathbf{w}^{k+1} = \begin{cases} \mathbf{w}^k + \mathbf{d}^k & \text{if } \rho_k > \eta_0, \\ \mathbf{w}^k & \text{if } \rho_k \le \eta_0, \end{cases} \tag{46}$$

where $\eta_0 > 0$ is a pre-specified value. The size Δ_k of the trust region is then updated according to the reduction of the function value. If the reduction is significant, then Δ_k is enlarged. Otherwise, we reduce Δ_k. More details can be found in Lin and Moré (1999). The framework of our trust region method is given in Algorithm 6. Earlier works applying trust region methods for L2-regularized problems include, for example, Lin et al. (2008). The algorithm here is more complicated due to the bound constraints.

5.1.1 CAUCHY POINT

A challenge for minimizing bound-constrained problems is to quickly identify bounded components at an optimal solution (i.e., components which are upper- or lower-bounded). Because each w_j can be either bounded or free, the number of combinations is exponential. One commonly used approach takes the negative gradient direction to obtain a new point and projects it back to the feasible region Ω. With a proper line search to ensure the reduction of the quadratic model $q_k(\mathbf{d})$, not only do we effectively guess the set of bounded components at an optimum, but also the convergence is guaranteed. To be more precise, we find a step size $\lambda > 0$ so that

$$q_k(\mathbf{d}^{k,C}) \le q_k(\mathbf{0}) + \sigma \nabla q_k(\mathbf{0})^T \mathbf{d}^{k,C} \quad \text{and} \quad \|\mathbf{d}^{k,C}\| \le \Delta_k, \tag{47}$$

where

$$\mathbf{d}^{k,C} = P[\mathbf{w}^k - \lambda \nabla \bar{f}(\mathbf{w}^k)] - \mathbf{w}^k \tag{48}$$

is called the Cauchy step in bound-constrained optimization and $\sigma \in (0, 1/2)$ is a constant. The projection operator $P[\cdot]$ maps $\mathbf{w}^k - \lambda \nabla \bar{f}(\mathbf{w}^k)$ back to the feasible region Ω:

$$P[w_j] = \min\left(u_j,\ \max(w_j, l_j)\right), \tag{49}$$

so some components become bounded. Although the negative gradient direction is projected in (48), the resulting direction is still a descending one (i.e., $\nabla q_k(\mathbf{0})^T \mathbf{d}^{k,C} < 0$). Hence, one can always find a small $\lambda > 0$ such that (47) is satisfied. The point $\mathbf{w}^{k,C} \equiv \mathbf{w}^k + \mathbf{d}^{k,C}$ is referred to as the Cauchy point.

5.1.2 NEWTON DIRECTION

Gradient descent methods suffer from slow convergence, so in (48) we should have used the Newton direction. However, the second-order information is accurate only if there are no bound constraints. Lin and Moré (1999) propose using Newton directions on the subspace of the Cauchy point's free components. Recall that we find the Cauchy point to predict the bounded elements at an optimum. We obtain the free/bounded sets at the Cauchy point

$$F \equiv F(\mathbf{w}^{k,C}) = \{j \mid l_j < w_j^{k,C} < u_j\} \quad \text{and} \quad B \equiv B(\mathbf{w}^{k,C}) = \{j \mid j \notin F\}, \tag{50}$$

and find a Newton direction on the space F by solving

$$\begin{aligned} \min_{\mathbf{v}_F} \quad & q_k(\mathbf{d}^{k,C} + \mathbf{v}) \\ \text{subject to} \quad & \|\mathbf{d}^{k,C} + \mathbf{v}\| \le \Delta_k,\ \mathbf{v}_B = \mathbf{0}. \end{aligned} \tag{51}$$

If the free set at the Cauchy point is close to that at an optimum, using a Newton direction on this sub-space leads to fast convergence.

Because (51) does not enforce the feasibility of $\mathbf{w}^{k,C} + \mathbf{v}$, one needs a projected line search procedure similar to (47)–(48). Details are shown later in (55). The resulting point may contain more bounded components than the Cauchy point. In this situation, Lin and Moré (1999) continue to minimize a quadratic approximation on the new sub-space. They thus generate inner iterates $\mathbf{w}^{k,1} = \mathbf{w}^{k,C}, \mathbf{w}^{k,2}, \mathbf{w}^{k,3}, \ldots$, until that the free/bounded sets do not change. If m inner iterations are taken, then the direction $\mathbf{d}^k$ for the kth trust region iteration in Algorithm 6 is

$$\mathbf{d}^k = \mathbf{w}^{k,m+1} - \mathbf{w}^k.$$

Details of our procedure are described in Algorithm 7. Because each inner iteration enlarges the bounded set, the number of inner iterations is bounded by $2n$, the number of variables. In practice, very few inner iterations (one or two) are taken. Another reason to take inner iterations is for the quadratic convergence proof in Lin and Moré (1999).

Let t be the inner-iteration index and B_t, F_t be bounded/free sets at $\mathbf{w}^{k,t}$. With $\mathbf{v}_{B_t} = \mathbf{0}$,

$$q_k(\mathbf{d}^{k,t}+\mathbf{v}) = \frac{1}{2}\mathbf{v}_{F_t}^T \nabla^2 \bar{f}(\mathbf{w}^k)_{F_t,F_t}\mathbf{v}_{F_t} + \nabla q_k(\mathbf{d}^{k,t})_{F_t}^T \mathbf{v}_{F_t} + q_k(\mathbf{d}^{k,t}), \tag{52}$$

so minimizing $q_k(\mathbf{d}^{k,t}+\mathbf{v})$ is equivalent to solving the following linear system

$$\nabla^2 \bar{f}(\mathbf{w}^k)_{F_t,F_t}\mathbf{v}_{F_t} = -\nabla q_k(\mathbf{d}^{k,t})_{F_t}. \tag{53}$$

To solve (53), we conduct conjugate gradient (CG) iterations until

$$\|\nabla^2 \bar{f}(\mathbf{w}^k)_{F_t,F_t}\mathbf{v}_{F_t} + \nabla q_k(\mathbf{d}^{k,t})_{F_t}\| = \|\nabla q_k(\mathbf{d}^{k,t+1})_{F_t}\| \le \varepsilon \|\nabla \bar{f}(\mathbf{w}^k)_{F_t}\| \tag{54}$$

is satisfied, where ε is a given positive constant. See Section 5.1.3 for reasons to choose CG. CG may stop before reaching (54) if either the iterate causes our search direction to exceed the trust region boundary or the singularity of the matrix $\nabla^2 \bar{f}(\mathbf{w}^k)_{F_t,F_t}$ is detected.

Once a direction $\mathbf{v}^{k,t}$ is identified, we conduct a projected line search to ensure the feasibility and the sufficient decrease of $q_k(\mathbf{d})$. This procedure is similar to (47)–(48) for the Cauchy step. We find λ (e.g., by a backtrack line search) such that

$$\begin{aligned} &\mathbf{w}^{k,t+1} = P[\mathbf{w}^{k,t} + \lambda \mathbf{v}^{k,t}],\ \mathbf{d}^{k,t+1} = \mathbf{w}^{k,t+1} - \mathbf{w}^k, \text{ and} \\ &q_k(\mathbf{d}^{k,t+1}) \le q_k(\mathbf{d}^{k,t}) + \sigma \nabla q_k(\mathbf{d}^{k,t})_{F_t}^T (\mathbf{d}^{k,t+1} - \mathbf{d}^{k,t})_{F_t}, \end{aligned} \tag{55}$$

where $P[\cdot]$ is defined in (49) and σ is the same as that in (47).

5.1.3 HESSIAN-VECTOR PRODUCT

CG is very suitable for solving the linear system (53) as it requires only Hessian-vector products. For logistic regression,

$$\nabla^2 \bar{f}(\mathbf{w})_{F_t,F_t} = C \begin{bmatrix} X^T \\ -X^T \end{bmatrix}_{F_t,:} D \begin{bmatrix} X & -X \end{bmatrix}_{:,F_t}, \tag{56}$$

where D is an $l \times l$ diagonal matrix with

$$D_{ii} = \tau\left(y_i(\mathbf{w}^+ - \mathbf{w}^-)^T \mathbf{x}_i\right)\left(1 - \tau(y_i(\mathbf{w}^+ - \mathbf{w}^-)^T \mathbf{x}_i)\right). \tag{57}$$

The matrix $\nabla^2 \bar{f}(\mathbf{w})_{F_t,F_t}$ may be too large to be stored. If using CG, the Hessian-vector product can be conducted by a sequence of matrix-vector products:

$$\nabla^2 \bar{f}(\mathbf{w}^{k,t})_{F_t,F_t}\mathbf{v}_{F_t} = C \begin{bmatrix} X^T \\ -X^T \end{bmatrix}_{F_t,:} \left(D\left(\begin{bmatrix} X & -X \end{bmatrix}_{:,F_t} \mathbf{v}_{F_t}\right)\right). \tag{58}$$

Thus, the memory problem is solved.

In Lin et al. (2008) for L2-regularized problems, Hessian-vector products are only needed in CG, but here they are also used in the projected line search for calculating $q_k(\mathbf{d}^{k,t+1}) - q_k(\mathbf{d}^{k,t})$; see (52) and (55). Moreover, in (58) we use only a sub-matrix of the Hessian, so $|F_t|$ columns of $\begin{bmatrix} X & -X \end{bmatrix}$ are needed. Because in general $|F_t|$ is smaller than the number of variables, calculating (58) may be faster than the product between the whole Hessian and a vector. To quickly access X's columns, storing the data matrix X in the column format is more suitable. For a discussion between row and column formats, see Lin et al. (2008, Section 4.3).

5.1.4 CONVERGENCE

From Theorem 2.1 of Lin and Moré (1999), any limit point of $\{\mathbf{w}^k\}$ is an optimum of (12). For the local convergence rate, in Appendix E, we indicate that in general TRON can achieve quadratic convergence.

5.2 An Interior Point Method (IPM)

Koh et al. (2007) proposed an interior point method to solve (13) with logistic loss. In (13), we omit the bias term b, but Koh et al. (2007) included this term. They consider a log barrier function so that $(\mathbf{w},\mathbf{u})$ is an interior point of the feasible region:

$$\phi_t(b,\mathbf{w},\mathbf{u}) \equiv t\left(\sum_{j=1}^{n} u_j + C\sum_{i=1}^{l}\xi(\mathbf{w},b;\mathbf{x}_i,y_i)\right) - \sum_{j=1}^{n}\log(u_j^2 - w_j^2),$$

where $t > 0$ is a parameter. The unique minimizer $(b^*(t),\mathbf{w}^*(t),\mathbf{u}^*(t))$ under any given t forms a curve called the "central path," which approaches an optimal solution of (13) as $t \to \infty$. An interior point method thus alternatively minimizes $\phi_t(b,\mathbf{w},\mathbf{u})$ and adjusts t. From a set of $(\mathbf{w},b)$ on the search path, we can construct a feasible solution to the dual problem of (13) and evaluate the duality gap. The duality gap is guaranteed to converge to 0 as we walk along the central path when $t \to \infty$. Thus, we can check the duality gap for the stopping condition.

At the kth iteration, using the current t_k, interior point methods approximately minimize $\phi_{t_k}(b,\mathbf{w},\mathbf{u})$ by finding a Newton direction. The following linear system is solved:

$$\nabla^2\phi_{t_k}(b^k,\mathbf{w}^k,\mathbf{u}^k)\begin{bmatrix}\Delta b\\ \Delta\mathbf{w}\\ \Delta\mathbf{u}\end{bmatrix} = -\nabla\phi_{t_k}(b^k,\mathbf{w}^k,\mathbf{u}^k). \tag{59}$$

For logistic loss,

$$\nabla\phi_t(b,\mathbf{w},\mathbf{u}) = \begin{bmatrix} tC\sum_{i=1}^{l} y_i\left(\tau(y_i(\mathbf{w}^T\mathbf{x}_i+b))-1\right)\\ tC\sum_{i=1}^{l}\left(\tau(y_i(\mathbf{w}^T\mathbf{x}_i+b))-1\right)y_i\mathbf{x}_i + \begin{bmatrix}2w_1/\left(u_1^2-w_1^2\right)\\ \vdots\\ 2w_n/\left(u_n^2-w_n^2\right)\end{bmatrix}\\ t\mathbf{e}_n - \begin{bmatrix}2u_1/\left(u_1^2-w_1^2\right)\\ \vdots\\ 2u_n/\left(u_n^2-w_n^2\right)\end{bmatrix}\end{bmatrix}$$

and

$$\nabla^2\phi_t(b,\mathbf{w},\mathbf{u}) = \begin{bmatrix} tC\mathbf{y}^TD\mathbf{y} & tC\mathbf{e}_l^TDX & \mathbf{0}^T\\ tCX^TD\mathbf{e}_l & tCX^TDX+D_1 & D_2\\ \mathbf{0} & D_2 & D_1\end{bmatrix},$$

where $\tau(\cdot)$ is defined in (9), $\mathbf{e}_n \in R^n$ and $\mathbf{e}_l \in R^l$ are the vectors of all ones, D is similar to (57) but includes b, and D_1 and D_2 are $n \times n$ diagonal matrices:

$$(D_1)_{jj} = 2\left(u_j^2+w_j^2\right)/\left(u_j^2-w_j^2\right)^2 \quad \text{and} \quad (D_2)_{jj} = -4u_jw_j/\left(u_j^2-w_j^2\right)^2.$$

Algorithm 8 IPM for L1-regularized logistic regression

1. Given b^1 and an interior point $(\mathbf{w}^1, \mathbf{u}^1)$. Let $t^1 = \frac{1}{Cl}$.
2. For $k = 1, 2, 3, \ldots$
   - Obtain a Newton direction $\begin{bmatrix} \Delta b \\ \Delta \mathbf{w} \\ \Delta \mathbf{u} \end{bmatrix}$ by solving (59).
   - A backtrack line search procedure to ensure the sufficient decrease of $\phi_{t_k}(\cdot)$
   - Update
   $$\begin{bmatrix} b^{k+1} \\ \mathbf{w}^{k+1} \\ \mathbf{u}^{k+1} \end{bmatrix} = \begin{bmatrix} b^k + \lambda \Delta b \\ \mathbf{w}^k + \lambda \Delta \mathbf{w} \\ \mathbf{u}^k + \lambda \Delta \mathbf{u} \end{bmatrix}.$$
   - Construct a dual feasible point and evaluate the duality gap η.
   - Set
   $$t^{k+1} = \begin{cases} \max\left(\mu \min(2n/\eta, t^k),\, t^k\right) & \text{if } \lambda \geq s_{\min}, \\ t^k & \text{otherwise,} \end{cases}$$
   where μ and $s_{\min}$ are constants.

Koh et al. (2007) apply preconditioned conjugate gradient methods (PCG) to solve (59) with diagonal preconditioning.

For the convergence, a backtrack line search procedure is needed to ensure the sufficient decrease of $\phi_{t_k}(\cdot)$. Koh et al. (2007) did not discuss details of their method's convergence. However, because interior point methods are a type of Newton methods, they often enjoy fast local convergence.

5.3 Lassplore Method

Liu et al. (2009) apply Nesterov's method (Nesterov, 2003) to solve (14). This method can handle (14) with and without the bias term. For simplicity, we do not consider the bias term. In addition to the sequence of iterations $\{\mathbf{w}^k\}$, for faster convergence, Nesterov's method uses another sequence of searching points $\{\mathbf{s}^k\}$, where

$$\mathbf{s}^k = \mathbf{w}^k + \beta_k(\mathbf{w}^k - \mathbf{w}^{k-1}),$$

for some positive parameter β_k. From $\mathbf{s}^k$, we obtain $\mathbf{w}^{k+1}$ by taking the negative gradient direction:

$$\mathbf{w}^{k+1} = \mathbf{s}^k - \lambda_k \nabla \bar{L}(\mathbf{s}^k),$$

where

$$\bar{L}(\mathbf{w}) \equiv \sum_{i=1}^{l} \xi_i(\mathbf{w}; \mathbf{x}_i, y_i)$$

is the objective function of (14) and λ_k is the step size. Note that $\bar{L}(\mathbf{w})$ is different from $L(\mathbf{w})$ defined in (7) because the penalty parameter C is not needed in (14). Liu et al. (2009) suggest to estimate β_k by

$$\beta_k = \frac{\gamma_k(1 - \alpha_{k-1})}{\alpha_{k-1}(\gamma_k + \frac{\alpha_k}{\lambda_k})}, \tag{60}$$

Algorithm 9 Lassplore for solving (14)

- Given $\mathbf{w}^0$ and $\mathbf{w}^1$, $\alpha_0 = 0.5, \lambda_1 > 0, \gamma_1 \geq 0$.
- For $k = 1, 2, 3, \ldots$
  1. While 1 do
     - Compute $\alpha_k \in (0,1)$ as the root of $\eta_k(\alpha)$, β_k by (60), and γ_{k+1} by (61).
     - Compute $\mathbf{s}^k = \mathbf{x}^k + \beta_k(\mathbf{w}^k - \mathbf{w}^{k-1})$.
     - Compute $\mathbf{w}^{k+1}$ by (63).
     - If $\bar{L}(\mathbf{w}^{k+1})$ satisfies (62)
       goto Step 2.
       Else
       $\lambda_k \leftarrow \lambda_k/2$.
  2. Find the initial λ_{k+1} for the next iteration by an adaptive scheme.[3]

where $\alpha_k \in (0,1)$ is the root of a quadratic function

$$\eta_k(\alpha) \equiv \frac{\alpha^2}{\lambda_k} + \gamma_k \alpha - \gamma_k$$

and $\gamma_k \geq 0$ satisfies

$$\gamma_{k+1} = (1 - \alpha_k)\gamma_k \text{ if } k \geq 1 \text{ and } \gamma_1 \geq 0. \tag{61}$$

We have $\alpha_k \in (0,1)$ because $\eta_k(0) = -\gamma_k < 0$ and $\eta_k(1) = 1/\lambda_k > 0$.

Lassplore applies an adaptive line search scheme that adjusts λ_k so that $\mathbf{w}^{k+1}$ satisfies

$$\bar{L}(\mathbf{w}^{k+1}) \leq \bar{L}(\mathbf{s}^k) + \nabla \bar{L}(\mathbf{s}^k)^T(\mathbf{w}^{k+1} - \mathbf{s}^k) + \frac{1}{2\lambda_k}\|\mathbf{w}^{k+1} - \mathbf{s}^k\|_2^2. \tag{62}$$

The new solution $\mathbf{w}^{k+1}$ generated from the above process may not satisfy the constraint in (14), so Lassplore projects the solution to the feasible region:

$$\mathbf{w}^{k+1} \equiv \arg\min_{\mathbf{w}} \{\|(\mathbf{s}^k - \lambda_k \nabla L(\mathbf{s}^k)) - \mathbf{w}\| \mid \|\mathbf{w}\|_1 \leq K\}. \tag{63}$$

This projection is related to (15) in Section 2.2.2, but Lassplore applies the method in Liu and Ye (2009) to efficiently compute (63).

A summary of Lassplore is given in Algorithm 9.

6. Three Other Methods for L1-regularized Logistic Regression

We describe details of three more methods because they are included in our comparison.

6.1 Orthant-Wise Limited-memory Quasi-Newton (OWL-QN)

LBFGS (Liu and Nocedal, 1989) is a limited memory quasi Newton method for unconstrained smooth optimization. It can not deal with (1) because of the non-differentiability. Andrew and Gao (2007) modified LBFGS to solve (1) and named their method as OWL-QN. Here, we discuss how they handle the non-differentiability.

3. See Liu et al. (2009, Algorithm 3) for details.

From earlier discussion, we know that (1) is differentiable in a region where the sign of w_j does not change. Thus, OWL-QN restricts the search space to such a region (called orthant). At the kth iteration, it searches $\mathbf{w}^{k+1}$ on the space:

$$\Omega_k \equiv \{\mathbf{w} \in R^n \mid \text{sgn}(w_j) = s_j^k, j = 1, \ldots, n\},$$

where

$$s_j^k \equiv \begin{cases} \text{sgn}(w_j^k) & \text{if } w_j^k \neq 0, \\ \text{sgn}(-\bar{\nabla}_j f(\mathbf{w}^k)) & \text{otherwise}, \end{cases} \tag{64}$$

and

$$\bar{\nabla}_j f(\mathbf{w}) \equiv \begin{cases} L'_j(\mathbf{w}) + 1 & \text{if } w_j > 0 \text{ or } (w_j = 0 \text{ and } L'_j(\mathbf{w}) + 1 < 0), \\ L'_j(\mathbf{w}) - 1 & \text{if } w_j < 0 \text{ or } (w_j = 0 \text{ and } L'_j(\mathbf{w}) - 1 > 0), \\ 0 & \text{otherwise} \end{cases} \tag{65}$$

is defined as the pseudo gradient of $f(\mathbf{w})$. In (64), if $w_j^k = 0$, we consider the space where w_j can be moved by taking the negative gradient direction. OWL-QN then approximately minimizes a quadratic approximation of (1) in the search space Ω_k:

$$\begin{aligned} \min_{\mathbf{d}} \quad & f(\mathbf{w}^k) + \bar{\nabla} f(\mathbf{w}^k)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T H_k \mathbf{d} \\ \text{subject to} \quad & \mathbf{w}^k + \mathbf{d} \in \Omega_k, \end{aligned} \tag{66}$$

where H_k approximates the Hessian of $f(\mathbf{w}^k)$ by the first-order information gathered from previous iterations. Details for getting H_k can be found in Liu and Nocedal (1989). To approximately solve (66), OWL-QN finds the minimum of the quadratic objective function:

$$\mathbf{d}^k = -H_k^{-1} \bar{\nabla} f(\mathbf{w}^k), \tag{67}$$

obtains a direction $\bar{\mathbf{d}}^k$ by confining $\mathbf{d}^k$ on the same orthant as $-\bar{\nabla} f(\mathbf{w}^k)$:

$$\bar{d}_j^k = \begin{cases} d_j^k & \text{if } \text{sgn}(d_j^k) = \text{sgn}(-\bar{\nabla}_j f(\mathbf{w}^k)), \\ 0 & \text{otherwise}, \end{cases} \tag{68}$$

and then conducts a backtracking line search to find λ such that the sufficient decrease of the function value is satisfied. Note that $\mathbf{w}^{k+1}$ must be in Ω_k, so following (64),

$$w_j^{k+1} = \begin{cases} w_j^k + \lambda \bar{d}_j^k & \text{if } \text{sgn}(w_j^k + \lambda \bar{d}_j^k) = s_j^k, \\ 0 & \text{otherwise}. \end{cases}$$

Algorithm 10 summarizes the procedure.

Regarding the convergence, Yu et al. (2010, Appendix D) point out that the proof by Andrew and Gao (2007) is flawed. Yu et al. (2010) give the convergence proof for a slightly modified algorithm.

Algorithm 10 OWL-QN: An extension of LBFGS for L1-regularized logistic regression

1. Given $\mathbf{w}^1$.
2. For $k = 1, 2, 3, \ldots$
   - Compute $\bar{\nabla} f(\mathbf{w}^k)$ by (65).
   - Obtain H^k by information gathered from previous iterations and compute $\mathbf{d}^k$ by (67).
   - Compute $\bar{\mathbf{d}}^k$ by (68).
   - Find $\mathbf{w}^{k+1} \in \Omega_k$ by a backtracking line search.

6.2 Generalized Linear Model with Elastic Net

Friedman et al. (2010) proposed GLMNET to handle least-square and log-linear losses with L1/L2 regularization. Here, we discuss how GLMNET solves L1-regularized logistic regression. Although GLMNET can solve (5) with the bias term, for simplicity, we only indicate how GLMNET solves (1).

Because of the twice differentiability of the logistic loss function, the gradient of $L(\mathbf{w})$ is shown in (42) and the Hessian is

$$\nabla^2 L(\mathbf{w}) = CX^T DX, \tag{69}$$

where $D \in R^{l \times l}$ is a diagonal matrix with

$$D_{ii} = \tau(y_i \mathbf{w}^T \mathbf{x}_i)\big(1 - \tau(y_i \mathbf{w}^T \mathbf{x}_i)\big) \tag{70}$$

and $\tau(\cdot)$ is defined in (9). See similar formulations derived earlier for TRON in (56) and (57). Given the current solution $\mathbf{w}^k$, GLMNET considers a quadratic approximation of $L(\mathbf{w})$. By the second-order Taylor expansion,

$$\begin{aligned} & f(\mathbf{w}^k + \mathbf{d}) - f(\mathbf{w}^k) \\ =& \Big(\|\mathbf{w}^k + \mathbf{d}\|_1 + L(\mathbf{w}^k + \mathbf{d}) \Big) - \Big(\|\mathbf{w}^k\|_1 + L(\mathbf{w}^k) \Big) \\ \approx& \nabla L(\mathbf{w}^k)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 L(\mathbf{w}^k) \mathbf{d} + \|\mathbf{w}^k + \mathbf{d}\|_1 - \|\mathbf{w}^k\|_1. \end{aligned}$$

Then, GLMNET solves the Newton-like system

$$\min_{\mathbf{d}} \quad q_k(\mathbf{d}) \equiv \nabla L(\mathbf{w}^k)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 L(\mathbf{w}^k) \mathbf{d} + \|\mathbf{w}^k + \mathbf{d}\|_1 - \|\mathbf{w}^k\|_1 \tag{71}$$

by a cyclic coordinate descent method. Following the framework in Algorithm 1, $\mathbf{d}$'s values are sequentially updated by minimizing the following one-variable function:

$$\begin{aligned} g_j(z) &\equiv q_k(\mathbf{d} + z\mathbf{e}_j) - q_k(\mathbf{d}) \\ &= |w_j^k + d_j + z| - |w_j^k + d_j| + Bz + \frac{1}{2} A z^2, \end{aligned}$$

where

$$B \equiv \nabla_j L(\mathbf{w}^k) + \sum_t \nabla^2_{jt} L(\mathbf{w}^k) d_t \quad \text{and} \quad A \equiv \nabla^2_{jj} L(\mathbf{w}^k)$$

can be calculated using (42) and (69). It is easy to minimize $g_j(z)$ by (27).[4]

4. In GLMNET implementation, instead of finding z, a different but equivalent update is used to get new $w_j^k + d_j$.

Algorithm 11 GLMNET for L1-regularized logistic regression

1. Given $\mathbf{w}^1$.
2. For $k = 1, 2, 3, \ldots$
   - Let $\mathbf{d}^k \leftarrow \mathbf{0}$.
   - While $\mathbf{d}^k$ is not optimal for minimizing $q_k(\mathbf{d})$
     - For $j = 1, \ldots, n$
       - Solve the following one-variable problem by (27):

$$\bar{z} = \arg\min_{z} \quad q_k(\mathbf{d}^k + z\mathbf{e}_j) - q_k(\mathbf{d}^k).$$

       - $d_j^k \leftarrow d_j^k + \bar{z}$.
   - $\mathbf{w}^{k+1} = \mathbf{w}^k + \mathbf{d}^k$.

Because calculating the matrix D involves many exponential operations, GLMNET also considers using an approximation of $\nabla^2 L(\mathbf{w})$ and minimizes

$$q_k(\mathbf{d}) \equiv \nabla L(\mathbf{w}^k)^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T H \mathbf{d} + \|\mathbf{w}^k + \mathbf{d}\|_1 - \|\mathbf{w}^k\|_1,$$

where $H \equiv 0.25CX^T IX$ and I is an identity matrix. That is, we use a cheaper but less accurate approximation of $f(\mathbf{w}^k + \mathbf{d}) - f(\mathbf{w}^k)$. A sketch of GLMNET is in Algorithm 11.

We briefly describe some GLMNET's implementation details. GLMNET applies a shrinking technique to solve a smaller optimization problem than (71); see similar techniques for decomposition methods in Section 4.1.2. Using a sparse representation of $\mathbf{w}$ and maintaining an index set Ω to indicate the non-zero elements of $\mathbf{d}$, GLMNET solves a smaller problem by a coordinate descent method:

$$\min_{\mathbf{d}_\Omega} \quad \nabla L(\mathbf{w}^k)^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T H \mathbf{d} + \|\mathbf{w}^k + \mathbf{d}\|_1 - \|\mathbf{w}^k\|_1.$$

GLMNET conducts feature-wise normalization before solving the optimization problem. That is, it solves (1) by replacing $\mathbf{x}_i$ with $\tilde{\mathbf{x}}_i$, where

$$\tilde{x}_{ij} \equiv \frac{x_{ij} - \bar{x}_j}{\sigma_j}, \forall i, j, \quad \bar{x}_j = \frac{\sum_{i=1}^{l} x_{ij}}{l}, \forall j, \quad \text{and } \sigma_j = \sqrt{\sum_{i=1}^{l} x_{ij}^2 - \sum_{i=1}^{l} \bar{x}_j^2}, \forall j.$$

Notice that there is no guarantee that GLMNET converges to an optimum. Furthermore, the function value may not decrease because GLMNET does not conduct a line search procedure on the direction $\mathbf{d}$. For minimizing the quadratic approximation $q_k(\mathbf{d})$, GLMNET measures the relative step change in the successive coordinate descent iterations. Deciding when to stop minimizing $q_k(\mathbf{d})$ is an issue because a strict stopping condition may already cause long running time for $q_1(\mathbf{d})$.

6.3 Bundle Method

Bundle method is a cutting plane approach for minimizing non-smooth convex problems. Teo et al. (2010) proposed a bundle method BMRM to handle non-differentiable loss functions (e.g., L1 loss). They provide an extension to handle L1 regularization. Interestingly, BMRM applies cutting planes only to the loss function, regardless of whether it is differentiable or not. Therefore, for problem (1)

Algorithm 12 BMRM for L1-regularized logistic regression

1. Given $\mathbf{w}^1$.
2. For $k = 1, 2, 3, \ldots$
   - Compute and store $\mathbf{a}_k$ and b_k by (73).
   - Obtain $\mathbf{w}^{k+1}$ by solving the linear program (76).

with logistic loss, BMRM uses a non-smooth method to handle the smooth loss function and some other ways for the non-smooth regularization term $\|\mathbf{w}\|_1$.

Let $\mathbf{w}^k$ be the solution at the kth iteration. Using the convexity of the loss function, BMRM builds a cutting plane (i.e., the first-order Taylor expansion) of $L(\mathbf{w})$ at $\mathbf{w} = \mathbf{w}^k$:

$$\begin{aligned} L(\mathbf{w}) \geq & \nabla L(\mathbf{w}^k)^T(\mathbf{w} - \mathbf{w}^k) + L(\mathbf{w}^k) \\ = & \mathbf{a}_k^T \mathbf{w} + b_k, \ \forall \mathbf{w}, \end{aligned} \tag{72}$$

where

$$\mathbf{a}_k \equiv \nabla L(\mathbf{w}^k) \quad \text{and} \quad b_k \equiv L(\mathbf{w}^k) - \mathbf{a}_k^T \mathbf{w}^k. \tag{73}$$

If $L(\mathbf{w})$ is non-differentiable, in (72), BMRM substitutes $\nabla L(\mathbf{w})$ with the sub-gradient of $L(\mathbf{w})$.

BMRM maintains all cutting planes from the earlier iterations to form a lower-bound function for $L(\mathbf{w})$:

$$L(\mathbf{w}) \geq L_k^{\mathrm{CP}}(\mathbf{w}) \equiv \max_{1 \leq t \leq k} \mathbf{a}_t^T \mathbf{w} + b_t, \ \forall \mathbf{w}. \tag{74}$$

BMRM obtains $\mathbf{w}^{k+1}$ by solving the following sub-problem:

$$\min_{\mathbf{w}} \quad \|\mathbf{w}\|_1 + L_k^{\mathrm{CP}}(\mathbf{w}). \tag{75}$$

Using (74) and the splitting of $\mathbf{w}$ in (12) by $\mathbf{w} = \mathbf{w}^+ - \mathbf{w}^-$, Equation (75) can be reformulated to the following linear programming problem:

$$\begin{aligned} \min_{\mathbf{w}^+, \mathbf{w}^-, \zeta} \quad & \sum_{j=1}^{n} w_j^+ + \sum_{j=1}^{n} w_j^- + \zeta \\ \text{subject to} \quad & \mathbf{a}_t^T (\mathbf{w}^+ - \mathbf{w}^-) + b_t \leq \zeta, \quad t = 1, \ldots, k, \\ & w_j^+ \geq 0, \ w_j^- \geq 0, \quad j = 1, \ldots, n. \end{aligned} \tag{76}$$

A summary of the BMRM approach is given in Algorithm 12. Teo et al. (2010, Appendix C) indicated that because of the L1 regularization, the convergence result has not been fully established yet.

7. L1-regularized L2-loss Support Vector Machines

Previous sections have focused on L1-regularized logistic regression. Now we consider (1) with the L2-loss function (4). The optimization problem can be rewritten as

$$\min_{\mathbf{w}} \quad f(\mathbf{w}) \equiv \|\mathbf{w}\|_1 + C \sum_{i \in I(\mathbf{w})} b_i(\mathbf{w})^2, \tag{77}$$

where

$$b_i(\mathbf{w}) \equiv 1 - y_i \mathbf{w}^T \mathbf{x}_i \quad \text{and} \quad I(\mathbf{w}) \equiv \{i \mid b_i(\mathbf{w}) > 0\}. \tag{78}$$

Therefore, the sum of losses is

$$L(\mathbf{w}) = C \sum_{i \in I(\mathbf{w})} b_i(\mathbf{w})^2. \tag{79}$$

In contrast to logistic loss, the L2-loss function is differentiable but not twice differentiable (Mangasarian, 2002). Thus, some methods discussed in Sections 4–6 may not be directly applicable because they use second-order information. However, as shown in Mangasarian (2002, Section 3), (79) is twice differentiable at all but $\{\mathbf{w} \mid b_i(\mathbf{w}) = 0 \text{ for some } i\}$. Moreover, $\nabla L(\mathbf{w})$ is globally Lipschitz continuous, so a generalized Hessian exists everywhere. Using a generalized Hessian, we may modify algorithms using the second-order information for L1-regularized L2-loss SVMs. In the following two sections, we extend CDN and TRON to solve (77).

7.1 CDN for L2-loss SVMs

To apply CDN for L2-loss SVMs, in the sub-problem (17), we have

$$L_j(z) = C \sum_{i \in I(\mathbf{w}^{k,j} + z\mathbf{e}_j)} b_i(\mathbf{w}^{k,j} + z\mathbf{e}_j)^2.$$

For a second-order approximation similar to (26), we need $L_j'(0)$ and $L_j''(0)$:

$$L_j'(0) = -2C \sum_{i \in I(\mathbf{w}^{k,j})} y_i x_{ij} b_i(\mathbf{w}^{k,j}).$$

Unfortunately, $L_j''(0)$ is not well-defined if there exists some i such that $b_i(\mathbf{w}^{k,j}) = 0$. Following Chang et al. (2008), we consider the generalized second derivative:

$$2C \sum_{i \in I(\mathbf{w}^{k,j})} x_{ij}^2. \tag{80}$$

By replacing $L_j''(0)$ in (26) with the above quantity, we can easily obtain a direction d. However, the value in (80) may be zero if $x_{ij} = 0,\ \forall i \in I(\mathbf{w}^{k,j})$. To apply the convergence result in Tseng and Yun (2007), we ensure the strict positivity by taking

$$\max\left(2C \sum_{i \in I(\mathbf{w}^{k,j})} x_{ij}^2,\ \varepsilon\right), \tag{81}$$

where ε is a small positive value. Following the explanation in Appendix F, we have the finite termination of the line search procedure and the asymptotic convergence of the function value.

Like the situation for logistic regression, the major cost for finding the Newton direction d and for the line search procedure is to calculate $\mathbf{w}^T \mathbf{x}_i,\ \forall i \in I$. We maintain $b_i(\mathbf{w}),\ \forall i$ using the same trick in (25). All other implementation techniques discussed in Section 4.1.2 for logistic regression can be applied here.

7.2 TRON for L2-loss SVMs

We apply TRON to solve the bound-constrained problem (12) using L2 loss. Following the notation in Section 5.1, $\mathbf{w} \in R^{2n}$ is defined in (43) and $\bar{f}(\mathbf{w})$ is the objective function in (12).

The quadratic model $q_k(\mathbf{d})$ requires the gradient and the Hessian of $\bar{f}(\mathbf{w})$. We have

$$\nabla \bar{f}(\mathbf{w}) = \mathbf{e} + 2C\left(\bar{X}_{I,:}^T \bar{X}_{I,:}\mathbf{w} - \bar{X}_{I,:}^T \mathbf{y}_I\right),$$

where $\mathbf{e} \in R^{2n}$ is a vector of all ones, $\bar{X} \equiv \begin{bmatrix} X & -X \end{bmatrix}$, and I is defined in (78). $\bar{X}_{I,:}$ denotes a sub-matrix including $\bar{X}$'s rows corresponding to I; see (8). Note that $b_i(\mathbf{w})$ defined in (78) is now calculated by

$$1 - y_i(\mathbf{w}_{1:n} - \mathbf{w}_{(n+1):2n})^T \mathbf{x}_i.$$

Following Mangasarian (2002) and Fan et al. (2008, Appendix D), we consider the generalized Hessian matrix:

$$2C\bar{X}^T D \bar{X},$$

where $D \in R^{l \times l}$ is a diagonal matrix with

$$D_{ii} = \begin{cases} 1 & \text{if } b_i(\mathbf{w}) > 0, \\ 0 & \text{if } b_i(\mathbf{w}) \le 0. \end{cases}$$

We then apply Algorithm 7 to approximately minimize $q_k(\mathbf{d})$. For the Hessian-vector product, only instances in the index set I and variables in the free set F are considered. Thus, the Hessian-vector product in (58) becomes

$$2C\bar{X}_{I,F}^T \left(D_{I,I}\left(\bar{X}_{I,F}\mathbf{v}_F\right)\right),$$

where $\mathbf{v}$ is a vector in R^{2n}.

Regarding the convergence, from Theorem 2.1 of Lin and Moré (1999), any limit point of $\{\mathbf{w}^k\}$ is an optimal solution. However, without the twice differentiability, it is unclear if the local quadratic convergence holds.

8. Numerical Experiments

In Sections 4–7, we have described details of several large-scale optimization methods for L1-regularized linear classification. In this section, we conduct experiments to investigate their performances. We describe data sets and experimental settings first. Then, we comprehensively compare methods for logistic regression and L2-loss SVMs. Programs used in this paper are available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear/exp.html`.

8.1 Data Sets

Table 1 lists data statistics. Most data sets include documents, where the numbers of both features and instances are large. An exception is the problem a9a from UCI "adults" data sets; it has $l \gg n$. For other document sets, real-sim includes some Usenet articles, news20 is a collection of news documents, rcv1 is an archive of manually categorized newswire stories from Reuters, and yahoo-japan/yahoo-korea are document data from Yahoo!. Except yahoo-japan and yahoo-korea, other data sets are publicly available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/`

Data set	l	n	#nz
a9a	32,561	123	451,592
real-sim	72,309	20,958	3,709,083
news20	19,996	1,355,191	9,097,916
rcv1	677,399	47,236	49,556,258
yahoo-japan	176,203	832,026	23,506,415
yahoo-korea	460,554	3,052,939	156,436,656

Table 1: Statistics of data sets: l and n denote the numbers of instances and features in a data set, respectively. The column #nz indicates the number of non-zero entries.

Data set	LR without bias			LR with bias			L2-loss SVM		
	C	Density	Acc.	C	Density	Acc.	C	Density	Acc.
a9a	4.0	94.3	85.2	2.0	89.3	85.3	0.5	90.2	85.2
real-sim	4.0	16.8	97.1	4.0	16.2	97.0	1.0	18.9	97.1
news20	64.0	0.2	95.1	64.0	0.2	94.9	64.0	0.7	96.1
rcv1	4.0	23.8	97.8	4.0	23.0	97.8	1.0	25.9	97.8
yahoo-japan	4.0	1.3	91.9	4.0	1.0	93.1	1.0	1.4	92.2
yahoo-korea	4.0	1.0	87.6	4.0	0.9	87.7	1.0	1.0	87.7

Table 2: The best parameter C, the model density (%), and the testing accuracy (%). We conduct five-fold cross validation on the training set to select C in $\{2^k \mid k = -4, -3, \ldots, 6\}$. Using the selected C, we build a model to predict the testing set.

`datasets/`. For every document data set, instance-wise normalization has been conducted so that the length of each instance is one.

To estimate the testing accuracy, a stratified selection is taken to split each data set into one fifth for testing and the rest for training.

8.2 Experimental Settings

We consider the following implementations discussed in Sections 4–7.

- BBR: the cyclic coordinate descent method for logistic regression is described in Section 4.1.1. We download version 4.03 from `http://www.bayesianregression.org/`.
- CDN: the cyclic coordinate descent methods for logistic regression and L2-loss SVMs are respectively described in Sections 4.1.2 and 7.1. To check the sufficient decrease condition, we use $\sigma = 0.01$ and $\beta = 1/2$. For L2-loss SVM, we use $\varepsilon = 10^{-12}$ in (81). The implementation is the same as that included in version 1.6 of our software LIBLINEAR (`http://www.csie.ntu.edu.tw/~cjlin/liblinear/`).
  In Section 4.1.2, we discuss the shrinking technique for CDN. We defer the investigation of its effectiveness to Section 8.4. In all other places of the comparison, we run CDN with the shrinking strategy.
- SCD: the stochastic coordinate descent method for logistic regression is described in Section 4.1.3. The source code is available at `http://ttic.uchicago.edu/~tewari/code/scd/`.

- CGD-GS: the block coordinate gradient descent method for logistic regression is described in Section 4.2. Following Yun and Toh's (2009) settings, we choose

$$H = \text{diag}\left[\min\left(\max(\nabla^2_{jj} L(\mathbf{w}^k), 10^{-10}), 10^{10}\right)\right]_{j=1,\ldots,n}$$

  and apply the Gauss-Southwell-r rule to choose the working set J; see Equation (40). Note that from Table 9 of Yun and Toh (2009), implementations using Gauss-Southwell-r and Gauss-Southwell-q rules perform similarly on document data. We use default values for all parameters; see Section 5.1 in Yun and Toh (2009). In particular, $\sigma = 0.1$, $\beta = 1/2$, and $\gamma = 0$. The source code is available at `http://www.math.nus.edu.sg/~matys/`.
- TRON: the trust region Newton methods for logistic regression and L2-loss SVMs are respectively described in Sections 5.1 and 7.2. In the projected line search (47) and (55), we use $\sigma = 0.01$ and $\beta = 0.1$. For the stopping condition of the CG procedure in Algorithm 7, we use $\varepsilon = 0.1$. The parameter η_0 in (46) is 0.0001.
- IPM: the interior point method for logistic regression is described in Section 5.2. For small and dense data sets, IPM solves a linear system in Algorithm 8 by Cholesky factorization. For large and sparse data sets, it applies a preconditioned conjugate gradient method to approximately solve the linear system. In the experiment, we only consider the large and sparse setting. We use default parameters: $\sigma = 0.01$ and $\beta = 1/2$ for the line search procedure. To update t^k, we use $s_{\min} = 1/2$ and $\mu = 2$. The source code (version 0.8.2) is downloaded from `http://www.stanford.edu/~boyd/l1_logreg/`.

  As $\mathbf{w}$ lies in the interior of the feasible region, every w_j is non-zero after IPM stops. To gain the sparsity, following the condition (31), Koh et al. (2007) assign those w_j satisfying $|\nabla_j L(\mathbf{w})| \leq 0.9999$ to zero. However, if we have not run enough iterations to obtain an accurate solution, this modification of $\mathbf{w}$ may result in an erroneous model. We thus add another condition $|w_j| < 1$ in deciding if w_j should be assigned to zero. We will address this issue again in Section 8.3.

  We find that because of a problem of not initializing an element in an array, the previous version (0.8.1) of IPM is two or three times slower than the latest version (0.8.2). This observation indicates the difficulty in comparing software. Some minor issues may significantly affect the conclusions.
- OWL-QN: the quasi Newton method is described in Section 6.1. The source code (version 1.1.2) is available at `http://research.microsoft.com/en-us/um/people/jfgao/`.
- GLMNET: the method is described in Section 6.2 for logistic regression. Following the default setting, we use the full Hessian in (71) instead of an approximation. We gradually reduce the stopping tolerance to obtain different models. The coordinate descent method for minimizing the quadratic approximation $q_k(\mathbf{d})$ stops according to a tolerance the same as the overall stopping tolerance. The source code (version 1.5) is available at `http://cran.r-project.org/web/packages/glmnet/index.html`.
- Lassplore: this method is described in Section 5.3. We check the 1-norm of the optimal solution (obtained by other solvers) to calculate the value K in (14). The source code (version 1.0) is available at `http://www.public.asu.edu/~jye02/Software/lassplore`.
- BMRM: this bundle method is described in Section 6.3. We apply it to both logistic regression and L2-loss SVMs. The linear programming problem (76) is solved via GNU Linear Programming Kit (GLPK). The source code (version 2.2) and GLPK are available at `http://users.rsise.anu.edu.au/~chteo/BMRM.html` and `http://www.gnu.org/software/glpk/`, respectively.

GLMNET is implemented in Fortran along with an R interface. CGD-GS and Lassplore are primarily implemented in MATLAB, but expensive operations are coded in C/C++. All other solvers are implemented in C/C++ with double precision.

Some implementations (SCD, TRON, OWL-QN, and BMRM) solve (1), while some (CGD-GS, IPM, GLMNET, and Lassplore) consider the bias term and solve (5). BBR[5] and our CDN implementation can handle both (1) and (5). According to whether the bias term is considered, we categorize methods into two groups for comparison.[6] Moreover, for certain solvers their formulations are scaled by a constant. We ensure that equivalent optimization problems are solved.

Software such as IPM and GLMNET supports finding a solution path of various C values. However, in our experiments, we focus on the performance of an algorithm using a fixed C. For all methods, we set the initial solution $\mathbf{w}^1 = \mathbf{0}$.

The parameter C is chosen by five-fold cross validation (CV) on the training set. Using models trained under the best C, we predict the testing set to obtain the testing accuracy. The best C, the model density (the number of non-zero coefficients in $\mathbf{w}$ divided by n), and the corresponding testing accuracy are recorded in Table 2.

In the rest of this section, we compare the training speed of solvers for logistic regression and L2-loss SVMs by using the best parameter C of each data set. We run all experiments on a 64-bit machine with Intel Xeon 2.0GHz CPU (E5504), 128KB L1 cache, 1GB L2 cache, and 32GB main memory. We use GNU C/C++/Fortran compilers (version 4.4.1) and ensure that for each package the "-O3" optimization flag is set.

8.3 Comparing Methods for L1-regularized Logistic Regression

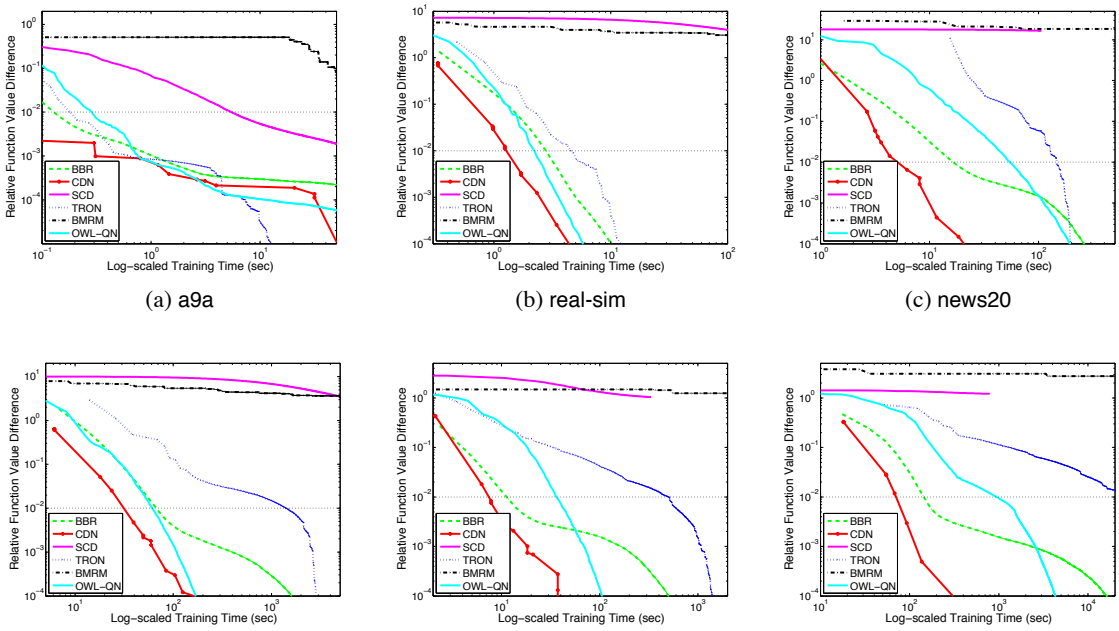
We separately compare solvers for optimization problems without/with the bias term. The first group, which solves (1), includes BBR, CDN, SCD, TRON, OWL-QN, and BMRM. We begin at showing in Figure 1 the relation between the function value and the training time. In each figure, the x-axis is the training time and the y-axis is the relative difference to the optimal function value:

$$\frac{f(\mathbf{w}) - f^*}{f^*}, \tag{82}$$

where f^* is obtained by running TRON with a strict stopping condition. Both x-axis and y-axis are log-scaled. We draw a dotted reference line in Figure 1 to indicate the relative error 0.1. From Figure 1, BBR and CDN can more quickly give a good solution than SCD, TRON, OWL-QN, and BMRM. However, quasi Newton and Newton methods may have faster local convergence. In Figures 1(c) and 1(d), TRON's curve is almost vertical in the end. This fast local convergence is mainly useful for problems such as a9a, for which BBR and CDN are less competitive. We note that a9a is not a document data set and its number of features is much smaller than the number of instances. Earlier studies for L2-regularized classifiers have shown that coordinate descent methods are less competitive for this type of data (Hsieh et al., 2008). The same situation seems to occur here for L1 regularization.

It is surprising that SCD is much slower than CDN and BBR because they are all coordinate descent methods. We modify CDN to randomly select working variables, a stochastic method similar to SCD; the result was still much faster than SCD, suggesting that the stochastic selection is not the

5. BBR considers bias term as a feature and allows users to use an option "-l" to specify weights to individual features.
6. For simplicity, we apply BBR only to solve (1).



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 1: Relative difference between the objective function and the optimum value, versus training time. Logistic regression without using the bias term. Both x-axis and y-axis are log-scaled.

culprit. It turns out that a too large upper bound of the second derivative causes the slow convergence of SCD. When $x_{ij} \in [-1,1]$, SCD replaces $\sum_{i=1}^{l} x_{ij}^2$ in (34) with l. Then its U_j in (35) is much larger than U_j in (22) for BBR. Therefore, SCD's step size $-g_j'(0)/U_j$ is often too small.

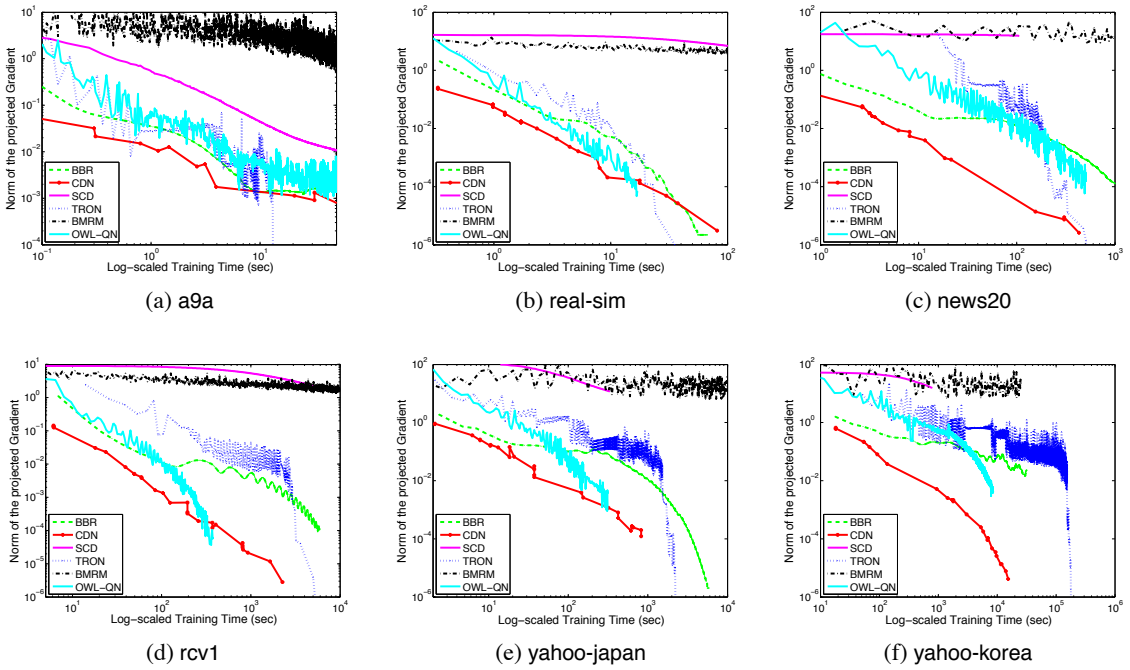
Equation (82) cannot be used for a practical stopping condition because f^* is not easily available. Instead, we often rely on the gradient information. Now (1) is not differentiable, so we follow the optimality condition (6) to define the minimum-norm sub-gradient:

$$\nabla_j^S f(\mathbf{w}) \equiv \begin{cases} \nabla_j L(\mathbf{w}) + 1 & \text{if } w_j > 0, \\ \nabla_j L(\mathbf{w}) - 1 & \text{if } w_j < 0, \\ \operatorname{sgn}(\nabla_j L(\mathbf{w})) \max(|\nabla_j L(\mathbf{w})| - 1, 0) & \text{otherwise.} \end{cases}$$

Since $\nabla^S f(\mathbf{w}) = \mathbf{0}$ if and only if $\mathbf{w}$ is optimal, we can check $\|\nabla^S f(\mathbf{w})\|$ for the stopping condition. Figure 2 shows the scaled 2-norm of the minimum-norm sub-gradient,

$$\frac{l}{\min(\sum_{i:y_i=1} 1, \sum_{i:y_i=-1} 1)\|\nabla^S f(\mathbf{w}^1)\|} \|\nabla^S f(\mathbf{w})\|, \tag{83}$$

along the training time. From Figure 2, CDN and BBR are the best in the early stage of the procedure, while Newton and quasi Newton methods such as TRON and OWL-QN have faster local convergence. This observation is consistent with Figure 1. Some methods (e.g., decomposition methods) do not calculate $\nabla^S f(\mathbf{w})$ in their procedures, so a gradient-based stopping condition may introduce extra cost. However, these methods may be able to use approximate gradient values obtained during the calculation. For example, coordinate descent methods calculate only $\nabla_j f(\mathbf{w}^{k,j}),\ j = 1,\ldots,n$



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

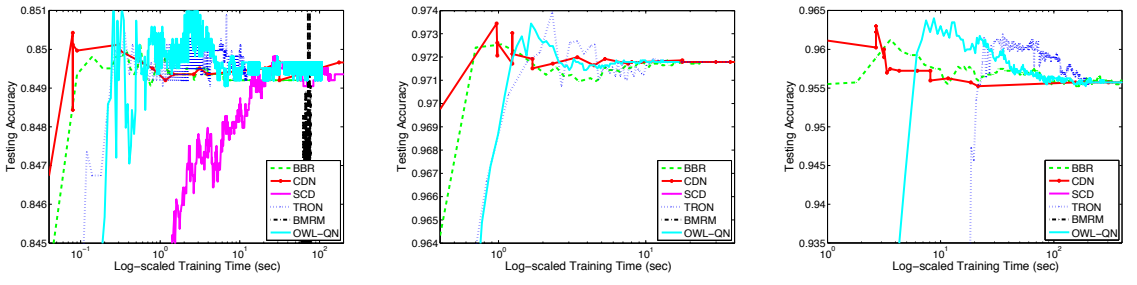
Figure 2: The 2-norm of the minimum-norm sub-gradient (83) versus training time. Logistic regression without using the bias term. Both x-axis and y-axis are log-scaled.

instead of $\nabla f(\mathbf{w}^{k+1})$, but these values can be directly used in a stopping condition; see details in Appendix F of Fan et al. (2008).

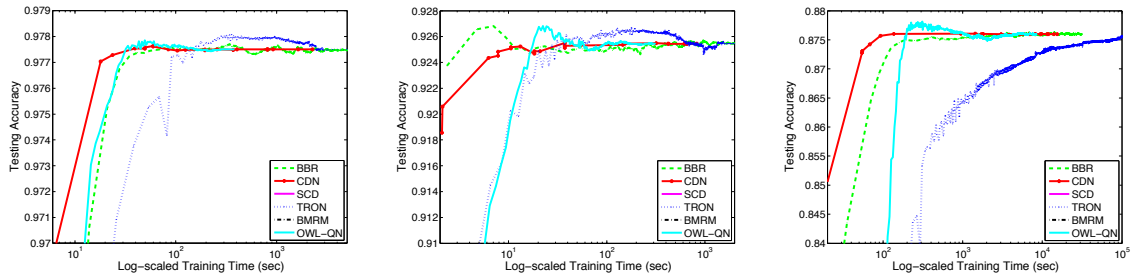
In Figure 3, we investigate how the testing accuracy is improved along the training time. The testing accuracy is the percentage of correct predictions on the testing set. Results show that BBR and CDN achieve the final accuracy more quickly. In addition, we are interested in the progress of these methods on gaining the sparsity. Figure 4 shows the number of non-zero coefficients of $\mathbf{w}$ and the training time. For most data sets, BBR and CDN more efficiently obtain a sparse solution.

Next, we compare methods solving (5) with the bias term. These solvers include CDN, CGD-GS, IPM, Lassplore, and GLMNET. Although BBR can solve (5), we omit it in the comparison as its performance is similar to that of solving (1). Following the comparison for methods solving (1), we begin with checking the running time to reduce the relative error of the function value and the scaled norm of the minimum-norm sub-gradient; see (82) and (83), respectively. The results are given in Figures 5 and 6. The reference value f^* is obtained using IPM with a strict stopping condition. From Figure 5, CDN is the fastest, IPM comes the second, and CGD-GS is the third. However, in Figure 6 for reducing the gradient norm, IPM often surpasses CDN in the final stage.

GLMNET gives good performances in Figures 5(b)–5(d). In particular, it has fast local convergence; see curves that are close to vertical in Figures 5(c) and 5(d). This fast local convergence is due to using the quadratic approximation $q_k(\mathbf{d})$, which generates a Newton-like direction. We find that the approximation of replacing D in (69) with a constant diagonal matrix is effective for micro-array data used in Friedman et al. (2010). However, for large document sets, using the exact Hessian is better. Unfortunately, GLMNET fails to generates results for yahoo-japan and yahoo-korea after sufficient running time. We find that setting an appropriate stopping tolerance for minimizing the

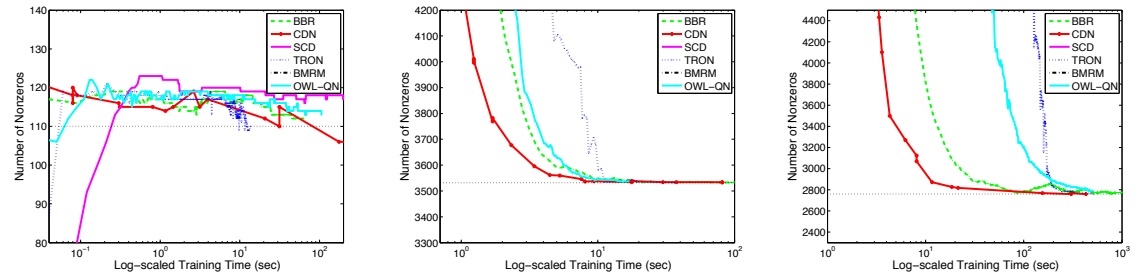


(a) a9a (b) real-sim (c) news20

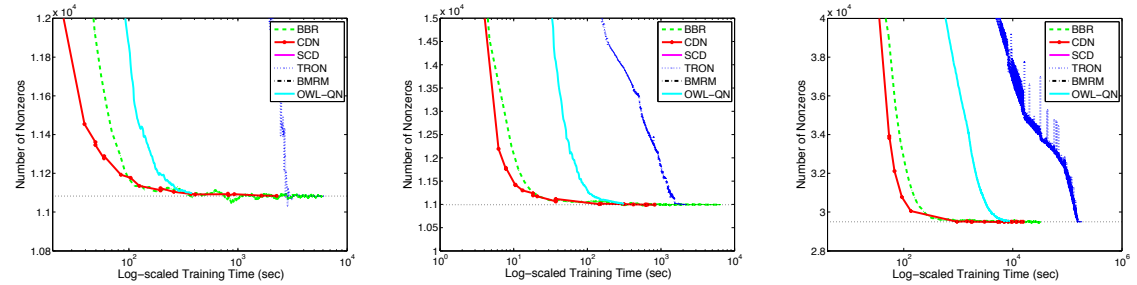


(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 3: The testing accuracy versus training time (log-scaled). Logistic regression without using the bias term. Curves of BMRM may not be shown because values are out of the range.

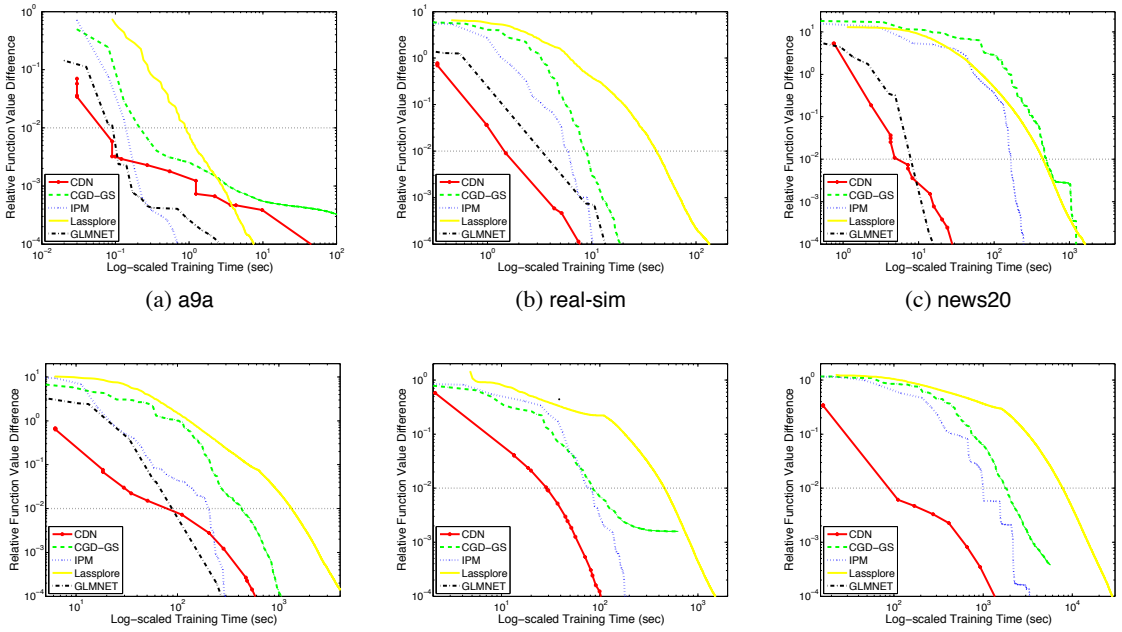


(a) a9a (b) real-sim (c) news20



(d) rcv1 (e) yahoo-japan (f) yahoo-korea

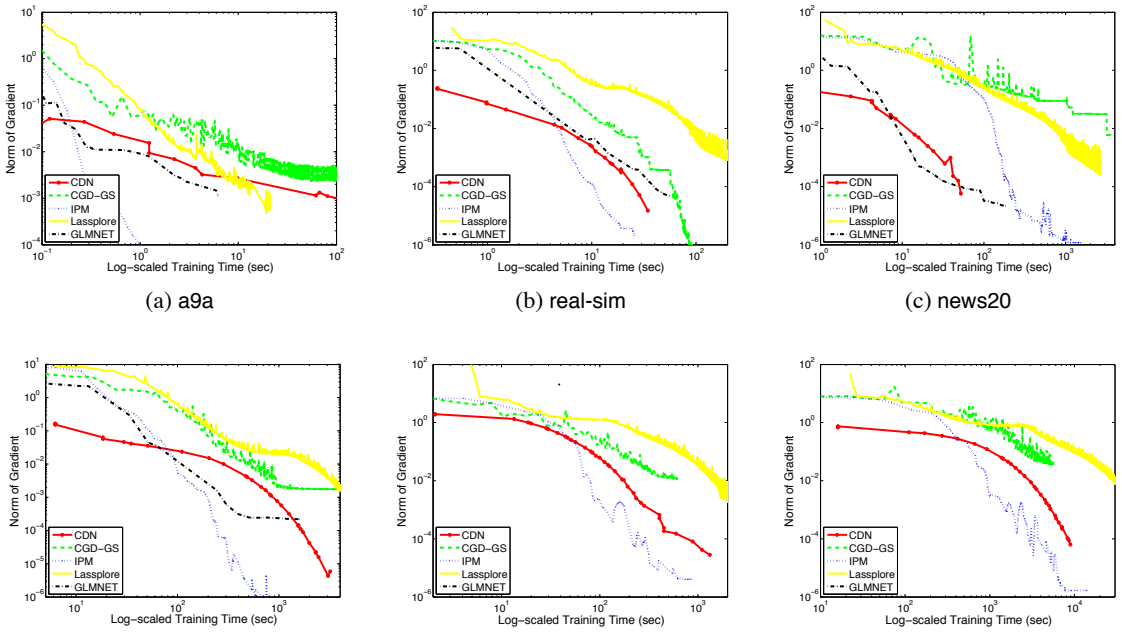
Figure 4: The number of non-zero coefficients versus training time (log-scaled). Logistic regression without using the bias term. Curves of BMRM may not be shown because values are out of the range. The solid horizontal line indicates the final number of non-zero coefficients.



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

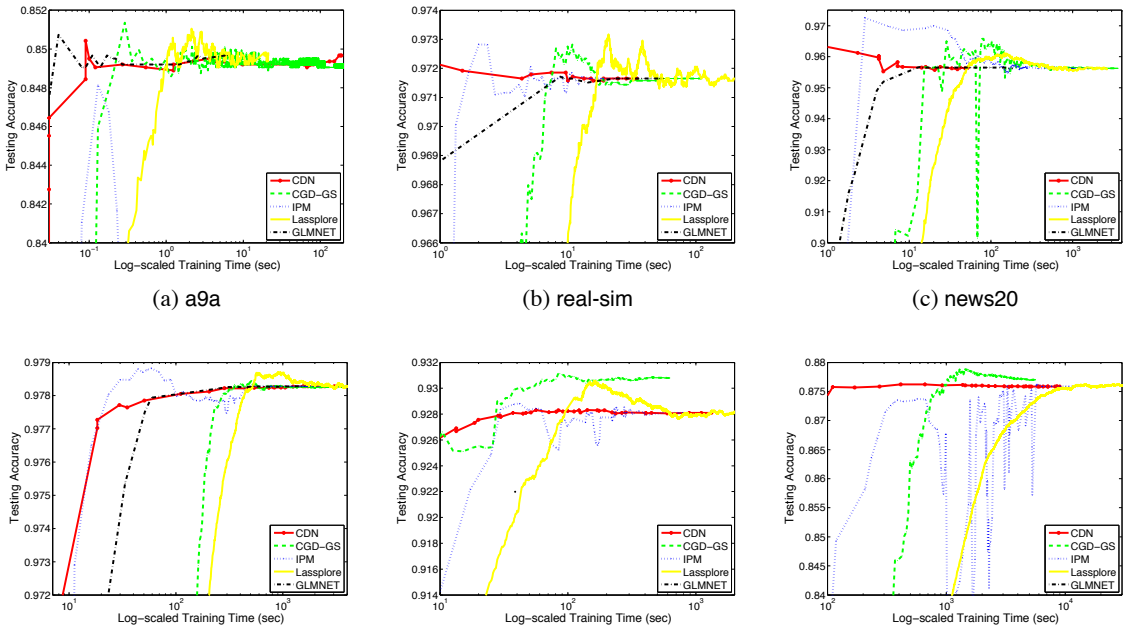
Figure 5: Relative difference between the objective function and the optimum value, versus training time. Logistic regression with the bias term. Both x-axis and y-axis are log-scaled.



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 6: The 2-norm of the minimum-norm sub-gradient (83) versus training time. Logistic regression with the bias term. Both x-axis and y-axis are log-scaled.



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

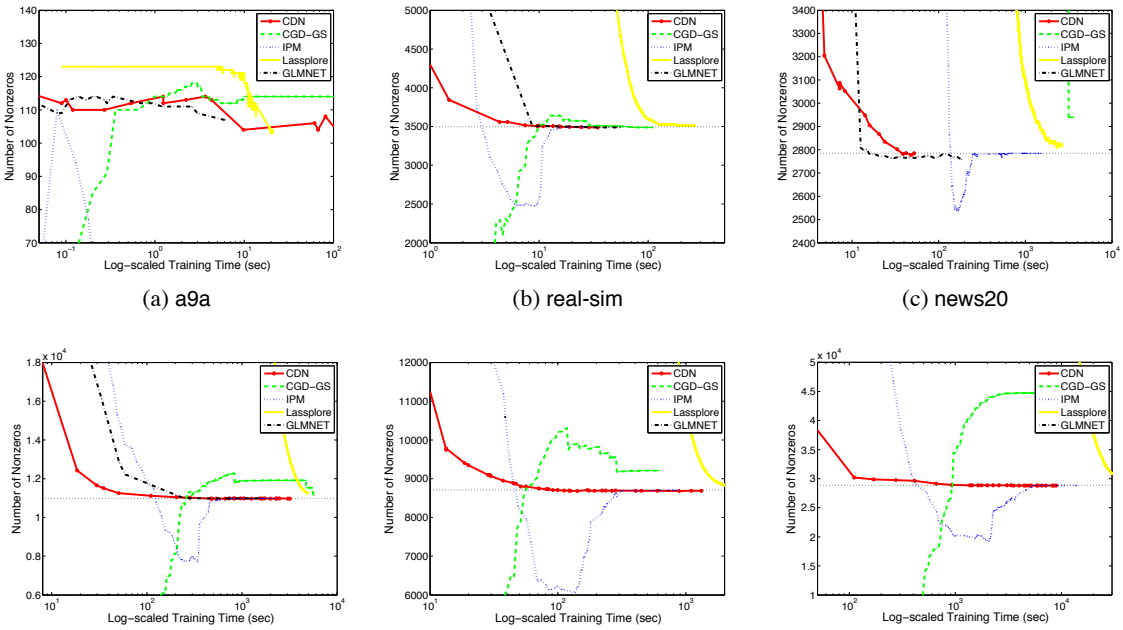
Figure 7: The testing accuracy versus training time (log-scaled). (log-scaled). Logistic regression with the bias term.

quadratic approximation in (71) is sometimes difficult. This issue might cause the problem in training yahoo-japan and yahoo-korea. For IPM, the performance is in general competitive. Because IPM is a type of Newton method, it has fast local convergence.

The result that CDN is faster than CGD-GS is worth for further discussion. Tseng and Yun (2007) show that for some least-square regression problems, an implementation similar to CDN (i.e., a Gauss-Seidel rule for selecting working variables) is slower than CGD-GS, which selects variables using the gradient information. This result is opposite to ours. Two issues might cause the different results. First, problems are different. We aim at classifying large and sparse document data, but they solve regression problems. Second, we implement two techniques (permutation of sub-problems and shrinking) to improve the convergence.

Yun and Toh (2009) show that for some document data sets used here, their CGD-GS is faster than IPM. However, our results indicate that IPM is better. This difference is apparently due to that Yun and Toh (2009) run an earlier version (0.8.1) of IPM. We indicate in Section 8.2 that a minor problem in this version causes this version to be two or three times slower than a later version (0.8.2) used here.

Figure 7 indicates the testing accuracy versus the training time. We can see in Figures 7(e) and 7(f) that IPM's accuracy may not improve as the training time increases. As we mentioned in Section 8.2, this result is because we modify certain $\mathbf{w}$ elements to zero. In the middle of the procedure, $\mathbf{w}$ is not close to an optimum yet, but many elements have satisfied $|\nabla_j L(\mathbf{w})| \leq 0.9999$ and are trimmed to zero. Hence, the resulting accuracy may be worse than that in the early stage of the procedure. In fact, due to IPM's dense $\mathbf{w}$ throughout iterations, to get the final sparsity and the testing accuracy, we need to accurately solve the optimization problem.



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 8: The number of non-zero coefficients versus training time (log-scaled). Logistic regression with the bias term. The solid horizontal line indicates the final number of non-zero coefficients.

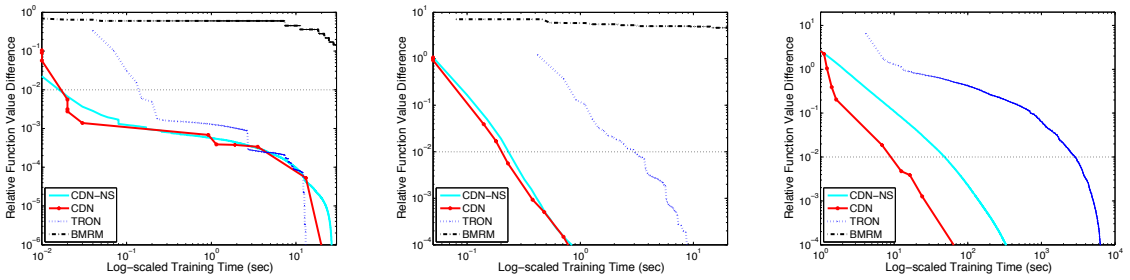
Figure 8 presents the number of $\mathbf{w}$'s non-zero coefficients. Similar to methods solving (1), all methods here, except CGD-GS, have solutions with many non-zero coefficients in the beginning and gradually gain the sparsity. In contrast, CGD-GS's numbers of non-zero coefficients in the whole optimization process are not much more than those of the final solutions. We find that this nice property is from using the gradient information for selecting working variables. In contrast, without the gradient information, CDN wrongly updates some elements of $\mathbf{w}$ to be non-zeros in the early stage of the procedure.

In summary, for large document data, coordinate descents methods such as CDN perform well in the early stage of the optimization procedure. As a result, they achieve the final accuracy more quickly. However, for some applications, a correct sparsity pattern of the model is important and a more accurate solution is sought. Then, GLMNET by combining both Newton-type and coordinate descent approaches is useful.

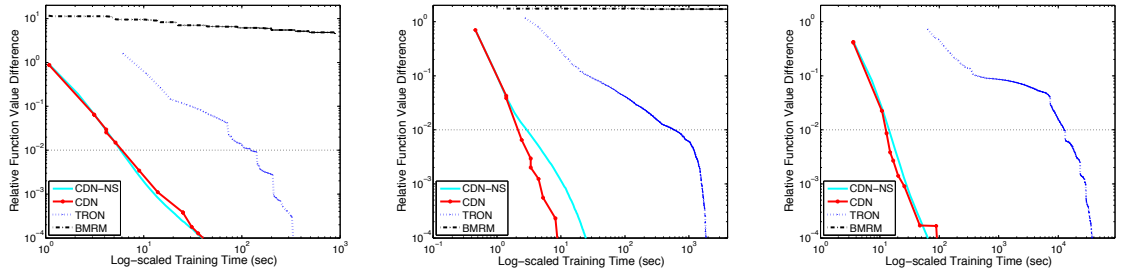
8.4 Comparing Methods for L1-regularized L2-loss SVMs and Investigating CDN's Shrinking Strategy

In Section 7, we have extended CDN and TRON to handle L2 loss, so methods included for comparison are CDN, TRON, and BMRM. Note that we solve (77) without considering the bias term.

Following the experiment for logistic regression, we plot the relative difference to the optimal function value in Figure 9 and the scaled norm of the minimum-norm sub-gradient in Figure 10. The reference f^* is obtained by running TRON with a strict stopping condition. One can see that



(a) a9a (b) real-sim (c) news20



(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 9: Relative difference between the objective function and the optimum value, versus training time. L2-loss SVMs without using the bias term. Both x-axis and y-axis are log-scaled.

CDN's training time is the shortest among the three solvers and TRON is better than BMRM. BMRM does not properly decrease the function value and the norm of gradient on large data sets.

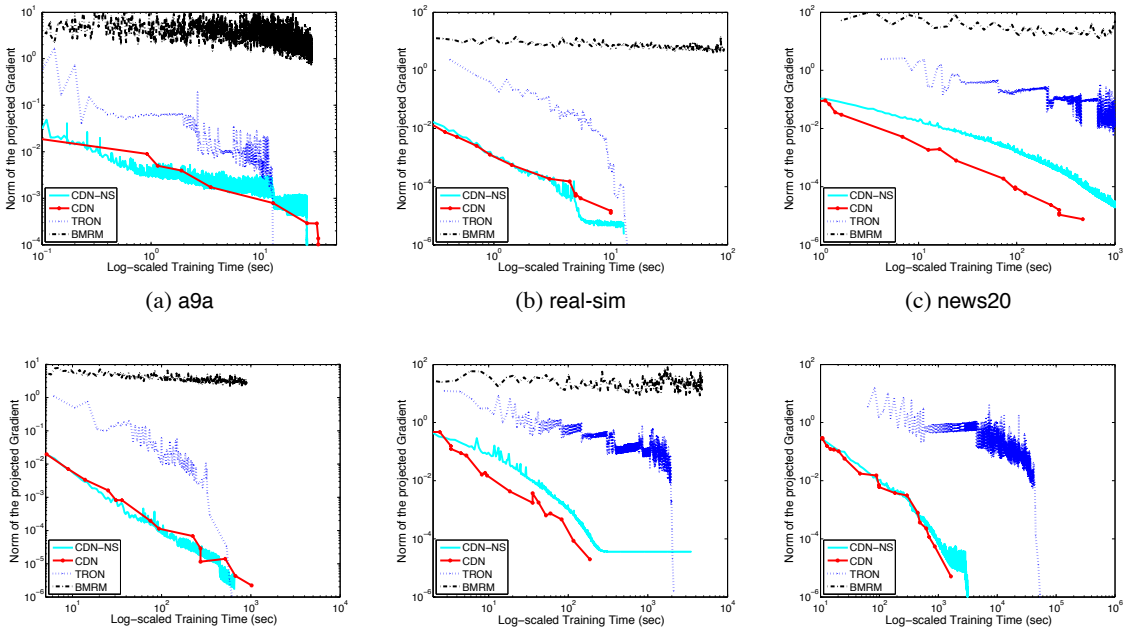
In Figures 9 and 10, we also present results of CDN without implementing the shrinking strategy (denoted as CDN-NS). In most cases, the implementation with shrinking is only slightly better. However, shrinking is effective if the sparsity is high (e.g., news20 and yahoo-japan). In such a situation, most $\mathbf{w}$ components are zero. We can safely remove some zero components and more efficiently solve smaller optimization problems.

9. Discussions and Conclusions

In Section 2.5, we briefly discuss optimization methods for L1-regularized least-square regression. Some comparisons can be seen in, for example, the experiment section of Wright et al. (2009) and Yun and Toh (2009). Note that an optimization method may perform differently on classification and regression problems. For instance, Yun and Toh (2009) show that CGD-GS is faster than CDN for regression problems, but here we have an opposite observation for document classification.

Figures 1–8 indicate that CDN is faster for solving (1) than (5). Our past work (Huang et al., 2010, Section 5) has indicated that the bias term may affect the running time of the same optimization method. As the accuracy does not differ much, it seems that in general the bias term should not be considered.

Among quasi Newton and Newton approaches, IPM and OWL-QN are faster than TRON. This result seems to indicate that TRON suffers from the difficulty for finding $\mathbf{w}$'s final non-zero coeffi-



(a) a9a (b) real-sim (c) news20

(d) rcv1 (e) yahoo-japan (f) yahoo-korea

Figure 10: The 2-norm of the minimum-norm sub-gradient (83) versus training time. L2-loss SVMs without using the bias term. Both x-axis and y-axis are log-scaled.

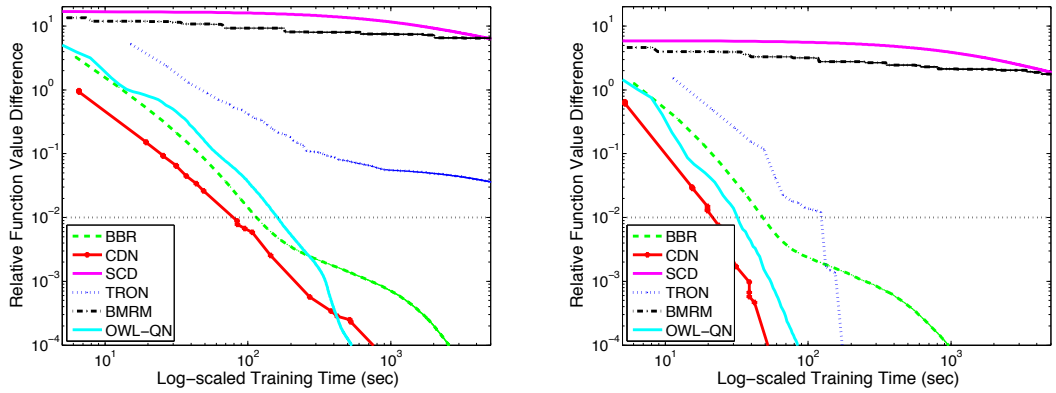
cients. However, IPM's $\mathbf{w}$ elements are all non-zeros, so we must accurately solve the optimization problem before trimming some elements to zero.

Figures 9 and 10 indicate that CDN/TRON for L2-loss SVMs is faster than CDN/TRON for logistic regression. Each iteration of CDN/TRON for L2-loss SVMs is cheaper because no exp/log operations are involved. Because the accuracy is similar, in general we prefer L2-loss SVMs over logistic regression.

The choice of the regularization parameter C affects the performance of solvers. In Figure 11, we present results on the rcv1 data set with regularization parameters $10C^*$ and $0.1C^*$, respectively, where $C^* = 4$ is the best parameter obtained from cross validation; see Table 2. Results show that all solvers take longer running time when C is large. Therefore, one should avoid a value much larger than C^* by trying from a small C. Moreover, methods using low-order (e.g., gradient only) information seem to be more sensitive to the change of C. For example, with C^* and $0.1C^*$ we respectively observe in Figures 1(d) and 11(b) that CDN is faster than OWL-QN, but with $10C^*$, OWL-QN surpasses CDN in the final stage.

GLMNET iteratively considers quadratic approximations and applies coordinate descent methods at each iteration. The discussion in Section 8.3 indicates that GLMNET's speed may be further improved if an adaptive stopping condition is properly designed for minimizing each quadratic approximation.

It is challenging to efficiently solve L1-regularized linear classification problems. The 1-norm term causes the non-differentiability of the objective function. In this paper, we review many existing methods for logistic regression and L2-loss SVMs. We discuss some state-of-the-art methods



(a) rcv1 with $10C^*$ (b) rcv1 with $0.1C^*$

Figure 11: A comparison of logistic regression solvers with regularization parameters $10C^*$ and $0.1C^*$. Both x-axis (relative difference between the objective function and the optimum value) and y-axis (training time) are log-scaled.

in detail. Our extensive comparison shows that carefully implemented coordinate descent methods are effective for L1-regularized classification with large-scale document data.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan via the grant 98-2221-E-002-136-MY3. The authors thank associate editor and reviewers for valuable comments.

Appendix A. Existence of Optimal Solutions of (1)

Consider the following level set:

$$S \equiv \{\mathbf{w} \mid f(\mathbf{w}) \leq f(\mathbf{0})\}.$$

We prove that S is compact (closed and bounded). Clearly, S is closed due to the continuity of $f(\mathbf{w})$. If S is not bounded, then there is a sequence $\{\mathbf{w}^k\} \subset S$ such that $\|\mathbf{w}^k\|_1 \to \infty$. Because we assume that $\xi(\mathbf{w};\mathbf{x}_i,y_i) \geq 0$, $f(\mathbf{w}^k) \geq \|\mathbf{w}^k\|_1$. Then, $f(\mathbf{w}^k) \to \infty$ contradicts $f(\mathbf{w}^k) \leq f(\mathbf{0})$. Thus, S is a compact set. From Weierstrass' Theorem, $f(\mathbf{w})$ has at least one minimum in S.

Appendix B. Solution of (26)

We consider the following general form with $A > 0$:

$$\min_z \quad |w_j + z| + Bz + \frac{1}{2}Az^2. \tag{84}$$

Clearly,

$$|w_j + z| + Bz + \frac{1}{2}Az^2 = \begin{cases} g_1(z) & \text{if } z \geq -w_j, \\ g_2(z) & \text{if } z \leq -w_j, \end{cases}$$

where

$$g_1(z) \equiv w_j + z + Bz + \frac{1}{2}Az^2 \quad \text{and} \quad g_2(z) \equiv -w_j - z + Bz + \frac{1}{2}Az^2.$$

By checking if the minimum of the quadratic function occurs on the right or the left side of $-w_j$, we know

$$\arg\min_{z \geq -w_j} g_1(z) = \begin{cases} -\frac{B+1}{A} & \text{if } B+1 \leq Aw_j, \\ -w_j & \text{otherwise,} \end{cases}$$

and

$$\arg\min_{z \leq -w_j} g_2(z) = \begin{cases} -\frac{B-1}{A} & \text{if } B-1 \geq Aw_j, \\ -w_j & \text{otherwise.} \end{cases}$$

Because $B+1 \leq Aw_j$ and $B-1 \geq Aw_j$ cannot hold at the same time, and $g_1(-w_j) = g_2(-w_j)$, we can easily conclude that the solution of (84) is

$$\begin{cases} -\frac{B+1}{A} & \text{if } B+1 \leq Aw_j, \\ -\frac{B-1}{A} & \text{if } B-1 \geq Aw_j, \\ -w_j & \text{otherwise.} \end{cases}$$

Appendix C. Proof of Theorem 1

From the assumption that $\mathbf{w}^k \to \mathbf{w}^*$ and the way that $\mathbf{w}$ is cyclically updated, we have $\mathbf{w}^{k,j} \to \mathbf{w}^*$ as well. The first result $-1 < \nabla_j L(\mathbf{w}^{k,j}) < 1$ immediately follows from the assumption $-1 < \nabla_j L(\mathbf{w}^*) < 1$ and the continuity of $\nabla L(\mathbf{w})$.

We then focus on the second result to show that $w_j^{k,j} = 0$ after k is large enough. The optimality condition (6) and the assumption $-1 < \nabla_j L(\mathbf{w}^*) < 1$ imply that $w_j^* = 0$. From the continuity of $\nabla^2 L(\mathbf{w})$ and the compactness of the level set S proved in Appendix A, we can define

$$M \equiv \max\{\nabla_{jj}^2 L(\mathbf{w}) \mid \mathbf{w} \in S\} \geq 0. \tag{85}$$

With the assumption $-1 < \nabla_j L(\mathbf{w}^*) < 1$ and the property $w_j^{k,j} \to w_j^* = 0$, for any $\sigma \in (0,1)$, there exists an iteration index K_j such that for all $k \geq K_j$,

$$-1 + \frac{M}{1-\sigma}|w_j^{k,j}| \leq L_j'(0; \mathbf{w}^{k,j}) = \nabla_j L(\mathbf{w}^{k,j}) \leq 1 - \frac{M}{1-\sigma}|w_j^{k,j}| \tag{86}$$

and

$$\left\{\mathbf{w} \;\middle|\; \|\mathbf{w} - \mathbf{w}^{k,j}\| \leq \|\mathbf{w}^{k,j} - \mathbf{w}^*\| \text{ for some } k \geq K_j\right\} \subset S.^7 \tag{87}$$

We prove that $w_j^{k,j} = 0$, $\forall k \geq K_j$. Recall that in the decomposition method we use (27) to obtain a direction d. From (86), we see that at $k = K_j$, the third case in (27) is taken. That is, $d = -w_j^{k,j}$. If σ in (86) is chosen to be the constant used in the sufficient decrease condition (28), we claim that

7. We need $f(\mathbf{w}^*) < f(\mathbf{0})$. This property generally holds. If not, we can slightly enlarge S so that (87) and all subsequent derivations still follow.

$d = -w_j^{k,j}$ satisfies (28) with $\lambda = 1$: If $w_j^{k,j} \geq 0$,

$$\begin{aligned} & g_j(-w_j^{k,j}) - g_j(0) - \sigma\left(-w_j^{k,j} - L_j'(0;\mathbf{w}^{k,j})w_j^{k,j}\right) \\ =&(1-\sigma)\left(-w_j^{k,j} - L_j'(0;\mathbf{w}^{k,j})w_j^{k,j}\right) + \frac{1}{2}L_j''(\tilde{z};\mathbf{w}^{k,j})(w_j^{k,j})^2 \qquad (88) \\ \leq&(1-\sigma)\left(-w_j^{k,j} - L_j'(0;\mathbf{w}^{k,j})w_j^{k,j}\right) + \frac{1}{2}M(w_j^{k,j})^2 \qquad (89) \\ \leq&\frac{-1}{2}M(w_j^{k,j})^2 \leq 0, \qquad (90) \end{aligned}$$

where $\tilde{z}$ is between 0 and $-w_j^{k,j}$, Equation (88) follows from (20)–(21), Equation (89) is from (87) and (85),[8] and Equation (90) is from the first inequality of (86). The above result is precisely the sufficient decrease condition (28) with $\lambda = 1$. The situation for $w_j^{k,j} < 0$ is similar. By taking the step $d = -w_j^{k,j}$, we have $w_j^{k+1,j} = 0$. At the $(k+1)$st iteration, $w_j^{k+1,j} = 0$ and (86) imply that the optimality condition (18) for the sub-problem has been satisfied. Thus, the jth element of $\mathbf{w}$ remains zero in all subsequent iterations.

Appendix D. Convergence of CDN: Logistic Regression

If each time one variable is used, the sub-problem considered in Tseng and Yun (2007) is

$$\min_z \quad |w_j + z| - |w_j| + L_j'(0)z + \frac{1}{2}Hz^2. \qquad (91)$$

Clearly, (26) is a special case of (91) by taking $H = L_j''(0)$. Therefore, we can apply results proved in Tseng and Yun (2007).

To have the finite termination of the line search procedure, Tseng and Yun (2007, Lemma 3.4) require that there exists $A > 0$ such that

$$\|\nabla L(\mathbf{w}_1) - \nabla L(\mathbf{w}_2)\| \leq A\|\mathbf{w}_1 - \mathbf{w}_2\|, \ \forall \mathbf{w}_1, \mathbf{w}_2 \in R^n \qquad (92)$$

and

$$L_j''(0;\mathbf{w}) > 0, \qquad (93)$$

where $\mathbf{w}$ is the current solution used to generate the direction d in (27). Equation (92) means that $L(\cdot)$ is globally Lipschitz continuous. For logistic regression, we have

$$\|\nabla L(\mathbf{w}_1) - \nabla L(\mathbf{w}_2)\| \leq \|\nabla^2 L(\tilde{\mathbf{w}})\|\|\mathbf{w}_1 - \mathbf{w}_2\|,$$

where $\tilde{\mathbf{w}}$ is between $\mathbf{w}_1$ and $\mathbf{w}_2$. Moreover,

$$\|\nabla^2 L(\tilde{\mathbf{w}})\| = C\|X^T D(\tilde{\mathbf{w}})X\| \leq C\|X^T\|\|X\|, \qquad (94)$$

where $D(\tilde{\mathbf{w}}) \in R^{l\times l}$ is a diagonal matrix similar to (70). Because any diagonal element of $D(\tilde{\mathbf{w}})$ is less than one, we have the inequality in (94). Thus, we can use $C\|X^T\|\|X\|$ as the constant A in (92).

8. More precisely,

$$\|\mathbf{w}^{k,j} + \tilde{z}\mathbf{e}_j - \mathbf{w}^{k,j}\| \leq |w_j^{k,j}| = |w_j^{k,j} - w_j^*| \leq \|\mathbf{w}^{k,j} - \mathbf{w}^*\|.$$

For (93), from (29), $L_j''(0;\mathbf{w})$ in the level set S is lower bounded by a positive value. The only exception is that $L_j''(0;\mathbf{w}) = 0$ when $x_{ij} = 0,\ \forall i = 1,\ldots,l$. In this situation, $L_j'(0;\mathbf{w}) = 0$ and the optimal $w_j^* = 0$. The one-variable function $g_j(z)$ is reduced to

$$g_j(z) = |w_j + z| - |w_j|,$$

so $d = -w_j$ from (27) and d satisfies the sufficient decrease condition (28). Therefore, w_j becomes zero after the first iteration and is not changed after. It is like that the jth feature has been removed in the beginning.

Next, we discuss the asymptotic convergence of function values. Tseng and Yun (2007) impose certain conditions on choosing the working set; see Equation (12) in their paper. If one variable is updated at a time, their condition is reduced to that between $\mathbf{w}^k$ and $\mathbf{w}^{k+1}$, one must go through n sub-problems covering all $w_1,\ldots,w_n$. This setting is exactly what we do, regardless of using a sequential order of $1,\ldots,n$ or a permutation $\pi(1),\ldots,\pi(n)$.

Tseng and Yun (2007) need an additional assumption for the asymptotic convergence (see Assumption 1 in their paper):

$$0 < \lambda_{\min} \le \nabla_{jj}^2 L(\mathbf{w}^{k,j}) \le \lambda_{\max}, \forall j = 1,\ldots,n, k = 1,\ldots, \tag{95}$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are positive constants. From (29) and the boundedness of the level set S (see Appendix A), this assumption holds except that for some $j, x_{ij} = 0, \forall i$. For such $j, \nabla_j L(\mathbf{w}) = 0, \forall \mathbf{w}$. Hence, w_j becomes zero by (27) at the first iteration and is not changed subsequently. Because $w_j = 0$ is optimal in this situation, it is like that the jth variable has been removed for optimization. Therefore, we have (95) without problems.

Following Tseng and Yun (2007, Theorem 4.1(e)), any limit point of $\{\mathbf{w}^k\}$ is an optimum of (1) with logistic loss.

Appendix E. Convergence of TRON: Logistic Regression

Consider any limit point $\bar{\mathbf{w}}$ generated by $\{\mathbf{w}^k\}$. Lin and Moré (1999) proved that if

$$\nabla^2 \bar{f}(\bar{\mathbf{w}})_{J,J} \text{ is positive definite,} \tag{96}$$

where

$$J \equiv \{j \mid \nabla_j \bar{f}(\bar{\mathbf{w}}) = 0\},$$

then the following two results hold:

1. $\{\mathbf{w}^k\}$ globally converges to $\bar{\mathbf{w}}$.
2. $\{\mathbf{w}^k\}$ quadratically converges. That is, there exist $\mu \in (0,1)$ and a positive integer K such that

$$\|\mathbf{w}^{k+1} - \bar{\mathbf{w}}\| \le (1-\mu)\|\mathbf{w}^k - \bar{\mathbf{w}}\|^2,\ \forall k > K.$$

The remaining question is whether (96) holds. From (58), we have

$$\nabla^2 \bar{f}(\bar{\mathbf{w}}) = C \begin{bmatrix} X^T \\ -X^T \end{bmatrix} D \begin{bmatrix} X & -X \end{bmatrix}, \tag{97}$$

where D is defined in (57). $\nabla^2 \bar{f}(\bar{\mathbf{w}})$ cannot be positive definite if there exists $1 \le j \le n$ such that $\{j, j+n\} \subset J$. The reason is that one can easily find a vector $\mathbf{v} \in R^{2n}$ such that $\mathbf{v}^T \nabla^2 \bar{f}(\bar{\mathbf{w}})\mathbf{v} = 0$. For

example, let $v_j = v_{j+n} \neq 0$ and other elements be zero. However, we claim that $\{j, j+n\} \subset J$ does not happen. Otherwise,

$$\nabla_j \bar{f}(\bar{\mathbf{w}}) = 1 + \nabla_j L(\bar{\mathbf{w}}_{1:n} - \bar{\mathbf{w}}_{n+1:2n}) \quad \text{and}$$
$$\nabla_{j+n} \bar{f}(\bar{\mathbf{w}}) = 1 - \nabla_j L(\bar{\mathbf{w}}_{1:n} - \bar{\mathbf{w}}_{n+1:2n})$$

are both zero, but this situation is not possible.

From the optimality condition (6), J is the same as the following set:

$$\{j \mid \bar{w}_j > 0 \text{ or } \bar{w}_j = \nabla_j \bar{f}(\bar{\mathbf{w}}) = 0\}.$$

Therefore, if the solution is sparse and those $\bar{w}_j = 0$ or $\bar{w}_{j+n} = 0, 1 \leq j \leq n$ satisfy

$$\begin{cases} \nabla_j \bar{f}(\bar{\mathbf{w}}) > 0 & \text{if } \bar{w}_j = 0, \\ \nabla_{j+n} \bar{f}(\bar{\mathbf{w}}) > 0 & \text{if } \bar{w}_{j+n} = 0, \end{cases}$$

(i.e., the optimization problem is non-degenerate), then $|J|$ is small. From (97), $\nabla^2 \bar{f}(\bar{\mathbf{w}})_{J,J}$ tends to be positive definite if $|J|$ is small. Then, we can have the quadratic convergence.

Appendix F. Convergence of CDN: L2-loss SVM

To have the finite termination of the line search procedure, we need conditions similar to (92) and (93). The condition (92) means that $\nabla L(\mathbf{w})$ is globally Lipschitz continuous. L2-loss SVM satisfies this property (Mangasarian, 2002, Section 3). For (93), the setting in (81) ensures that H in (91) is always positive. Therefore, the line search procedure stops in a finite number of steps.

For the asymptotic convergence, we need again the condition (95). Our setting in (81) meets this assumption, so any limit point of $\{\mathbf{w}^k\}$ is an optimal solution of (77).

References

Galen Andrew and Jianfeng Gao. Scalable training of L1-regularized log-linear models. In *Proceedings of the Twenty Fourth International Conference on Machine Learning (ICML)*, 2007.

Suhrid Balakrishnan and David Madigan. Algorithms for sparse linear classifiers in the massive data setting. 2005. URL `http://www.stat.rutgers.edu/~madigan/PAPERS/sm.pdf`.

Steven Benson and Jorge J. Moré. A limited memory variable metric method for bound constrained minimization. Preprint MCS-P909-0901, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 2001.

Paul S. Bradley and Olvi L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 1998.

Richard H. Byrd, Peihung Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16:1190–1208, 1995.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale L2-loss linear SVM. *Journal of Machine Learning Research*, 9:1369–1398, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cdl2.pdf`.

David L. Donoho and Yaakov Tsaig. Fast solution of l1 minimization problems when the solution may be sparse. *IEEE Transactions on Information Theory*, 54:4789–4812, 2008.

John Duchi and Yoram Singer. Boosting with structural sparsity. In *Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML)*, 2009.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

Mario Figueiredo, Robert Nowak, and Stephen Wright. Gradient projection for sparse reconstruction: Applications to compressed sensing and other inverse problems. *IEEE Journal on Selected Topics in Signal Processing*, 1:586–598, 2007.

Mário A. T. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1150–1159, 2003.

Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

Wenjiang J. Fu. Penalized regressions: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.

Glenn M. Fung and Olvi L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational optimization and applications*, 28:185–202, 2004.

Eli M. Gafni and Dimitri P. Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22:936–964, 1984.

Alexandar Genkin, David D. Lewis, and David Madigan. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

Joshua Goodman. Exponential priors for maximum entropy models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2004.

Elaine T. Hale, Wotao Yin, and Yin Zhang. Fixed-point continuation for l1-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19(3):1107–1130, 2008.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008. URL http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf.

Fang-Lan Huang, Cho-Jui Hsieh, Kai-Wei Chang, and Chih-Jen Lin. Iterative scaling and coordinate descent methods for maximum entropy. *Journal of Machine Learning Research*, 11:815–848, 2010. URL http://www.csie.ntu.edu.tw/~cjlin/papers/maxent_journal.pdf.

Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

Jun'ichi Kazama and Jun'ichi Tsujii. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 137–144, 2003.

S. Sathiya Keerthi and Shrish Shevade. A fast tracking algorithm for generalized LARS/LASSO. *IEEE Transactions on Neural Networks*, 18(6):1826–1830, 2007.

Jinseog Kim, Yuwon Kim, and Yongdai Kim. A gradient-based optimization algorithm for LASSO. *Journal of Computational and Graphical Statistics*, 17(4):994–1009, 2008.

Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior point method for large-scale l1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, 1:606–617, 2007.

Yongdai Kim and Jinseog Kim. Gradient LASSO for feature selection. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132:1–63, 1997.

Kwangmoo Koh, Seung-Jean Kim, and Stephen Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007. URL http://www.stanford.edu/~boyd/l1_logistic_reg.html.

Balaji Krishnapuram, Alexander J. Hartemink, Lawrence Carin, and Mario A. T. Figueiredo. A Bayesian approach to joint feature selection and classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):1105–1111, 2004.

Balaji Krishnapuram, Lawrence Carin, Mario A. T. Figueiredo, and Alexander J. Hartemink. Sparse multinomial logistic regression: fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, 2005.

Jussi Kujala, Timo Aho, and Tapio Elomaa. A walk from 2-norm SVM to 1-norm SVM. Preprint available from http://www.cs.tut.fi/~kujala2/papers/1norm2norm.pdf, 2009.

John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:771–801, 2009.

Cheng-Yu Lee. A comparison of optimization methods for large-scale l1-regularized logistic regression. Master's thesis, Department of Computer Science and Information Engineering, National Taiwan University, 2008.

Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y. Ng. Efficient l1 regularized logistic regression. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, pages 1–9, Boston, MA, USA, July 2006.

Chih-Jen Lin and Jorge J. Moré. Newton's method for large-scale bound constrained problems. *SIAM Journal on Optimization*, 9:1100–1127, 1999.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf`.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

Jun Liu and Jieping Ye. Efficient euclidean projections in linear time. In *Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML)*, 2009.

Jun Liu, Jianhui Chen, and Jieping Ye. Large-scale sparse logistic regression. In *Proceedings of The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 547–556, 2009.

Olvi L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

Olvi L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

Yurii E. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2003.

Michael R. Osborne, Brett Presnell, and Berwin A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20:389–404, 2000a.

Michael R. Osborne, Brett Presnell, and Berwin A. Turlach. On the LASSO and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000b.

Mee Young Park and Trevor Hastie. L1 regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society Series B*, 69:659–677, 2007.

Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.

Saharon Rosset. Following curved regularized optimization solution paths. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1153–1160, Cambridge, MA, 2005. MIT Press.

Volker Roth. The generalized LASSO. *IEEE Transactions on Neural Networks*, 15(1):16–28, 2004.

Sylvain Sardy, Andrew G. Bruce, and Paul Tseng. Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries. *Journal of Computational and Graphical Statistics*, 9:361–379, 2000.

Mark Schmidt, Glenn Fung, and Romer Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *Proceedings of European Conference on Machine Learning*, pages 286–297, 2007.

Mark Schmidt, Glenn Fung, and Romer Rosales. Optimization methods for l1-regularization. Technical Report TR-2009-19, University of British Columbia, 2009.

Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l1 regularized loss minimization. In *Proceedings of the Twenty Sixth International Conference on Machine Learning (ICML)*, 2009.

Shirish Krishnaj Shevade and S. Sathiya Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253, 2003.

Jianing Shi, Wotao Yin, Stanley Osher, and Paul Sajda. A fast hybrid algorithm for large scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 11:713–741, 2010.

Choon Hui Teo, S.V.N. Vishwanathan, Alex Smola, and Quoc V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B*, 58:267–288, 1996.

Ryota. Tomioka and Masashi Sugiyama. Dual augmented Lagrangian method for efficient sparse reconstruction. *IEEE Signal Processing Letters*, 16(12):1067–1070, 2009.

Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117:387–423, 2007.

Stephen J. Wright, Robert D. Nowak, and Mario A.T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57:2479–2493, 2009.

Tong Tong Wu and Kenneth Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

Jin Yu, S.V.N. Vishwanathan, Simon Gunter, and Nicol N. Schraudolph. A quasi-Newton approach to nonsmooth convex optimization problems in machine learning. *Journal of Machine Learning Research*, 11:1–57, 2010.

Sangwoon Yun and Kim-Chuan Toh. A coordinate gradient descent method for l1-regularized convex minimization. 2009. To appear in Computational Optimizations and Applications.

Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31, 2001.

Peng Zhao and Bin Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

Peng Zhao and Bin Yu. Stagewise lasso. *Journal of Machine Learning Research*, 8:2701–2726, 2007.

Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machines. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Journal of Machine Learning Research 11 (2010) 3235-3268 Submitted 7/09, Revised 2/10; Published 11/10

Approximate Riemannian Conjugate Gradient Learning for Fixed-Form Variational Bayes

Antti Honkela* ANTTI.HONKELA@TKK.FI
Tapani Raiko* TAPANI.RAIKO@TKK.FI
Mikael Kuusela MIKAEL.KUUSELA@TKK.FI
Matti Tornio MATTI.TORNIO@TKK.FI
Juha Karhunen JUHA.KARHUNEN@TKK.FI
Department of Information and Computer Science
Aalto University School of Science and Technology
P.O. Box 15400
FI-00076 AALTO, Finland

Editor: Manfred Opper

Abstract

Variational Bayesian (VB) methods are typically only applied to models in the conjugate-exponential family using the variational Bayesian expectation maximisation (VB EM) algorithm or one of its variants. In this paper we present an efficient algorithm for applying VB to more general models. The method is based on specifying the functional form of the approximation, such as multivariate Gaussian. The parameters of the approximation are optimised using a conjugate gradient algorithm that utilises the Riemannian geometry of the space of the approximations. This leads to a very efficient algorithm for suitably structured approximations. It is shown empirically that the proposed method is comparable or superior in efficiency to the VB EM in a case where both are applicable. We also apply the algorithm to learning a nonlinear state-space model and a nonlinear factor analysis model for which the VB EM is not applicable. For these models, the proposed algorithm outperforms alternative gradient-based methods by a significant margin.

Keywords: variational inference, approximate Riemannian conjugate gradient, fixed-form approximation, Gaussian approximation

1. Introduction

Bayesian methods have recently gained popularity in machine learning. This is at least partially due to their robustness against overfitting compared to maximum likelihood and other methods based on point estimates. Variational Bayesian (VB) methods provide an efficient and often sufficiently accurate deterministic approximation to Bayesian learning (Bishop, 2006). Mean field type VB also has the benefit that its objective function can be used for choosing the model structure or model order. Most work on variational methods has focused on the class of conjugate exponential models for which simple EM-like learning algorithms can be derived easily (Ghahramani and Beal, 2001; Winn and Bishop, 2005). These models and algorithms are computationally convenient but they rule out many interesting model types.

*. These authors contributed equally to this work. This project may be found at `http://www.cis.hut.fi/projects/bayes/`.

Many practically important models are not in the conjugate-exponential family and they have received far less attention in the VB literature. In this paper we present an efficient general method for applying VB learning to these more general models. The method could be used to speed up, for instance, the Gaussian variational approximation method of Opper and Archambeau (2009), or other previous more specific methods (e.g., Lappalainen and Honkela, 2000; Valpola and Karhunen, 2002).

Our method is based on first selecting the functional form of the approximation. For parts of the model that are conjugate-exponential, the corresponding factorised exponential family distribution is often a good choice, while in general we may use something else such as a multivariate Gaussian.

After fixing the functional form, we must be able to evaluate the variational free energy as a function of the variational parameters. The details of this step depend on the model—often most terms can be evaluated analytically while others may require computing some bounds (Jordan et al., 1999) or applying some linearisation techniques (see, e.g., Honkela and Valpola, 2005) or other approximations.

Once the free energy is known, we are left with a typically high-dimensional optimisation problem. Here[1] we propose using an approximate conjugate gradient algorithm that utilises the Riemannian geometry of the space of the approximations to speed up convergence. One of the main contributions of this paper is to use the geometry of the approximations. This is in contrast to more common applications of Riemannian geometry in natural gradient methods using the geometry of the predictive model. The geometry of the approximations is the natural choice if the variational inference is viewed as an optimisation problem in the space of approximating distributions. Furthermore, the geometry of the approximation is often much simpler, leading to more efficient computation and generic algorithms. The computational complexity of operations with the Fisher information matrix determining the geometry can be linear if the approximation is fully factorising or if its multivariate Gaussian blocks have a tree-like dependence structure, for instance. The resulting algorithm can provide dramatic speedups of potentially several orders of magnitude over state-of-the-art Euclidean conjugate gradient methods.

In previous machine learning algorithms Riemannian geometry is usually invoked through the natural gradient of Amari (1998). There, the aim has been to use maximum likelihood to directly update the model parameters $\boldsymbol{\theta}$ taking into account the geometry imposed by the predictive distribution of the data $p(\boldsymbol{X}|\boldsymbol{\theta})$. The resulting geometry is often quite complicated as the effects of different parameters cannot be separated and the Fisher information matrix is relatively dense. Recently Girolami and Calderhead (2011) have applied this in a Bayesian setting as a method to speed up Hamiltonian Monte Carlo samplers. In this paper, only the simpler geometry of the VB approximating distributions $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is used. Because the approximations are often chosen to minimise dependencies between different parameters $\boldsymbol{\theta}$, the resulting Fisher information matrix with respect to the variational parameters $\boldsymbol{\xi}$ will be mostly diagonal and hence easy to work with.

The rest of the paper is organised as follows. Section 2 introduces the background in variational Bayes and information geometry. The proposed approximate Riemannian conjugate gradient learning algorithm is introduced in Section 3. The method is demonstrated in three case studies: a Gaussian mixture model, a nonlinear state-space model and a nonlinear factor analysis model in Secs. 4, 5 and 6, respectively. We end with discussion in Section 7.

1. This paper is an extended version of the earlier conference paper (Honkela et al., 2008).

2. Background

The approximate Riemannian conjugate gradient learning algorithm follows very naturally from an optimisation view of variational Bayes and the Riemannian geometry of probability distributions in information geometry. We will start with a brief introduction to both of these techniques separately.

2.1 Variational Bayes

Variational Bayesian (VB) learning (Jordan et al., 1999; Ghahramani and Beal, 2001; Bishop, 2006) is based on approximating the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{X})$ with a tractable approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$, where $\boldsymbol{X}$ is the data, $\boldsymbol{\theta}$ are the unobserved variables (including both the parameters of the model and the latent variables), and $\boldsymbol{\xi}$ are the variational parameters of the approximation (such as the mean and the variance of a Gaussian approximation). The approximation is fitted by minimising the free energy

$$\mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi})) = E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left\{\log \frac{q(\boldsymbol{\theta}|\boldsymbol{\xi})}{p(\boldsymbol{X},\boldsymbol{\theta})}\right\} = D_{\mathrm{KL}}(q(\boldsymbol{\theta}|\boldsymbol{\xi})\|p(\boldsymbol{\theta}|\boldsymbol{X})) - \log p(\boldsymbol{X}). \tag{1}$$

This is equivalent to minimising the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(q\|p)$ between q and p (Bishop, 2006). The negative free energy also provides a lower bound on the marginal log-likelihood, that is, $\log p(\boldsymbol{X}) \geq -\mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}))$ due to non-negativity of the KL-divergence.

Typical classes of approximations used in VB include factorising approximations, most often starting from assuming the latent variables and parameters to be independent and extending to the fully factorising mean-field approximation, and approximations having a fixed functional form, such as a Gaussian.

In the former case, learning is typically performed using the variational Bayesian expectation maximisation (VB EM) algorithm which alternates between minimising the free energy with respect to the distribution of the latent variables (VB-E step) and the distribution of the parameters (VB-M step) (Bishop, 2006).

In the case of approximation with a fixed functional form, EM-like updates are usually not available and generic gradient-based optimisation methods have to be used (see, e.g., Opper and Archambeau, 2009). This is often very challenging in practice, as the problems are quite high dimensional and the lack of specific knowledge of interactions of the parameters that define the geometry of the problem can seriously hinder generic optimisation tools. Such methods have nevertheless been applied to a number of models that are not in the conjugate exponential family, such as multi-layer perceptron (MLP) networks (Barber and Bishop, 1998), kernel classifiers (Seeger, 2000), nonlinear factor analysis (Lappalainen and Honkela, 2000; Honkela and Valpola, 2005; Honkela et al., 2007) and nonlinear dynamical models (Valpola and Karhunen, 2002; Archambeau et al., 2008), non-conjugate variance models (Valpola et al., 2004; Raiko et al., 2007) as well as Gaussian process latent variable models (Titsias and Lawrence, 2010). Optimisation methods have included the conjugate gradient algorithm and heuristic speed-ups, but the use of a Riemannian conjugate gradient algorithm for VB as proposed in this paper is novel.

In practice, convergence of conjugate gradient algorithms in latent variable models is often really slow. To get an intuition of why this is the case, let us consider a generic latent variable model with latent variables that connect to one observation each and parameters that connect to a number of observations. As we increase the number of observations, the gradient with respect to the parameters grows linearly, whereas the gradient with respect to the latent variables stays constant.

As a result, with a reasonable number of observations, conjugate gradient algorithms are forced to take very small steps to avoid overshooting the parameters, and as a result the latent variables are hardly changed at all. The Riemannian gradient provides an automatic remedy to this problem by properly scaling the gradient.

2.2 Information Geometry and Optimisation on Riemannian Manifolds

When applying a generic optimisation algorithm to a problem such as optimising the free energy (1), a lot of background information on the geometry of the problem is lost. The parameters $\boldsymbol{\xi}$ of $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ can have different roles as location, shape, and scale parameters, and their effect is influenced by other parameters. This implies that the geometry of the problem is in most cases not Euclidean.

Riemannian geometry studies smooth manifolds $\{\mathcal{M}|\boldsymbol{\xi}\}$ that are locally diffeomorphic with $\mathbb{R}^n$. The manifold has an inner product $\langle\cdot,\cdot\rangle_k$ defined at every point $\boldsymbol{\xi}_k$ of the manifold. The inner product is defined for vectors in tangent space $T_{\boldsymbol{\xi}_k}$ of $\mathcal{M}$ at $\boldsymbol{\xi}_k$ as

$$\langle \mathbf{x}, \mathbf{y} \rangle_k = \mathbf{x}^T \mathbf{G}(\boldsymbol{\xi}_k) \mathbf{y} = \sum_{i,j} x_i y_j g_{ij}(\boldsymbol{\xi}_k), \tag{2}$$

where $\mathbf{G}(\boldsymbol{\xi}_k)$ is the Riemannian metric tensor at $\boldsymbol{\xi}_k$. Most Riemannian manifolds are curved, with geodesics as the counterparts of Euclidean straight lines as length-minimising curves between two points (Murray and Rice, 1993).

Information geometry studies the Riemannian geometric structure of the manifold of probability distributions (Amari, 1985). It has been applied to derive efficient natural gradient learning rules for maximum likelihood algorithms in independent component analysis (ICA) and multi-layer perceptron (MLP) networks (Amari, 1998). The approach has been used in several other problems as well, for example in analysing the properties of an on-line variational Bayesian EM method (Sato, 2001).

For probability distributions, the most natural metric is given by the Fisher information

$$g_{ij}(\boldsymbol{\xi}) = I_{ij}(\boldsymbol{\xi}) = E\left\{ \frac{\partial \log q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i} \frac{\partial \log q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_j} \right\} = E\left\{ -\frac{\partial^2 \log q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i \partial \xi_j} \right\}. \tag{3}$$

Here the last equality is valid if $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is twice continuously differentiable (Murray and Rice, 1993). Fisher information is a unique metric for probability distributions in the sense that it is the only metric which is invariant with respect to transformations to sufficient statistics (Amari and Nagaoka, 2000).

In a Riemannian space, the direction of steepest ascent of a function $\mathcal{F}(\boldsymbol{\xi})$ is given by the Riemannian or natural gradient

$$\tilde{\nabla}\mathcal{F}(\boldsymbol{\xi}) = \mathbf{G}^{-1}(\boldsymbol{\xi}) \nabla \mathcal{F}(\boldsymbol{\xi}) \tag{4}$$

instead of the regular gradient $\nabla\mathcal{F}(\boldsymbol{\xi})$. This can be verified by finding the maximum of $\mathcal{F}(\boldsymbol{\xi}+\Delta\boldsymbol{\xi})$ subject to the constraint $||\Delta\boldsymbol{\xi}||^2_{\boldsymbol{\xi}} = \langle \Delta\boldsymbol{\xi}, \Delta\boldsymbol{\xi} \rangle_{\boldsymbol{\xi}} = \Delta\boldsymbol{\xi}^T \mathbf{G}(\boldsymbol{\xi}) \Delta\boldsymbol{\xi} < \varepsilon^2$. The relation between gradient and Riemannian gradient is illustrated in Figure 1.

This choice of geometry between Euclidean and Riemannian is, however, independent of the choice of the optimisation algorithm, and recently several authors have combined conjugate gradient methods with the Riemannian or natural gradient (Smith, 1993; González and Dorronsoro, 2008; Honkela et al., 2008). In principle this can be achieved by replacing all vector space operations in

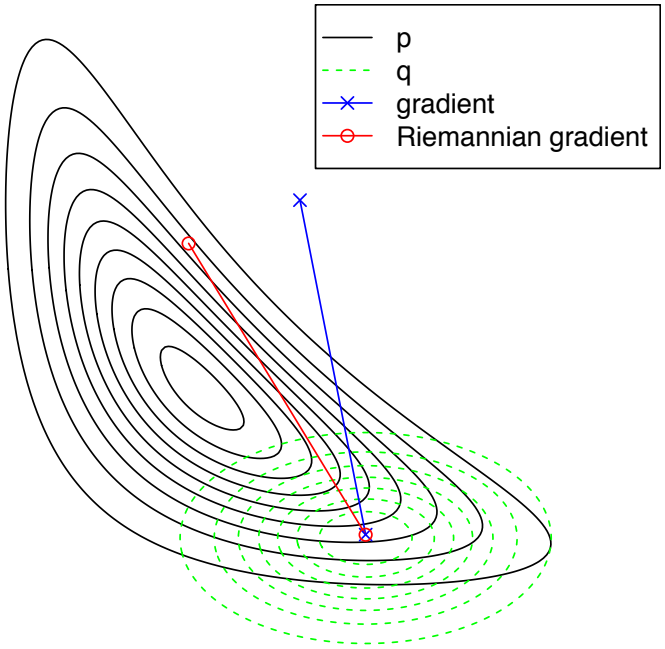


Figure 1: Gradient and Riemannian gradient directions are shown for the mean of distribution q. VB learning with a diagonal covariance is applied to the posterior $p(x,y) \propto \exp[-9(xy-1)^2 - x^2 - y^2]$. The Riemannian gradient strengthens the updates in the directions where the uncertainty is large.

the conjugate gradient algorithm with their Riemannian counterparts: Riemannian inner products and norms, parallel transport of gradient vectors between different tangent spaces as well as line searches and steps along geodesics in the Riemannian space. In practical algorithms some of these can be approximated by their flat-space counterparts. We shall apply the approximate Riemannian conjugate gradient (RCG) method which implements Riemannian (natural) gradients, inner products and norms but uses flat-space approximations of the others as our optimisation algorithm of choice throughout the paper. As shown in Appendix A, these approximations do not affect the asymptotic convergence properties of the algorithm. The difference between gradient and conjugate gradient methods is illustrated in Figure 2.

In this paper we propose using the Riemannian structure of the distributions $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ to derive more efficient algorithms for approximate inference and especially VB using approximations with a fixed functional form. This differs from the traditional natural gradient learning by Amari (1998) which uses the Riemannian structure of the predictive distribution $p(\boldsymbol{X}|\boldsymbol{\theta})$. The proposed method can be used to jointly optimise all the parameters $\boldsymbol{\xi}$ of the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$, or in conjunction with VB EM for some parameters.

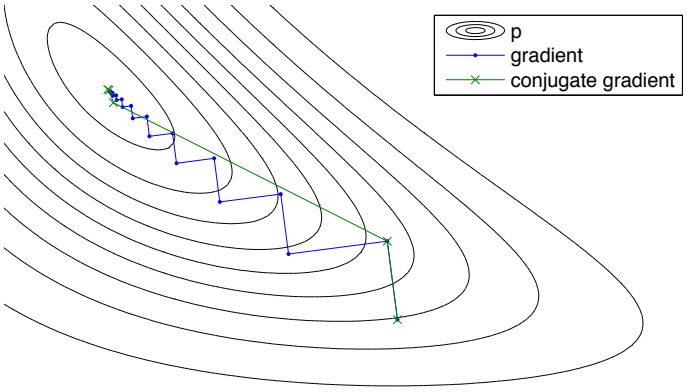


Figure 2: Gradient and conjugate gradient updates are applied to finding the maximum of the posterior $p(x,y) \propto \exp[-9(xy-1)^2 - x^2 - y^2]$. The step sizes that maximise p are used. Note that the first steps are the same, but following gradient updates are orthogonal whereas conjugate gradient finds a much better direction.

2.3 Information Geometry of VB EM

The optimal VB approximation has an information-geometric interpretation as a specific projection of the true posterior to a manifold of tractable distributions (Tanaka, 2001). This interpretation is equally valid for all optimisation methods.

Amari (1995) has also presented the geometric interpretation of the EM algorithm as alternating projections for E- and M-steps. This asymmetric view does not directly generalise to the VB EM method when used to infer distributions over all parameters, because VB EM is symmetric with respect to different parameters.

By embedding the VB-E step update within the VB-M step with point estimates and considering the resulting update, the VB EM algorithm for conjugate exponential family models can be interpreted as a natural gradient method (Sato, 2001). It therefore implicitly optimally utilises the Riemannian geometric structure of $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ (Amari, 1998). Nevertheless, the EM-based methods are prone to slow convergence, especially under low noise, even though more elaborate optimisation schemes can speed up their convergence somewhat. It is worth pointing out that this correspondence of VB EM is with the regular natural gradient algorithm, not Riemannian (natural) conjugate gradients as proposed in this paper.

3. Approximate Riemannian Conjugate Gradient Learning for Fixed-Form VB

Given a fixed-form approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ and the free energy $\mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}))$, it is possible to use standard gradient-based optimisation techniques to minimise the free energy with respect to $\boldsymbol{\xi}$. We will use VB EM updates for some variational parameters $\boldsymbol{\xi}^{\mathrm{EM}}$ and RCG for others $\boldsymbol{\xi}^{\mathrm{RCG}}$.

Instead of a regular Euclidean gradient algorithm, we optimise the free energy using a conjugate gradient algorithm that is adapted to Riemannian space by using Riemannian inner products and norms instead of Euclidean ones. Steps are still taken along Euclidean straight lines and the step

length is determined using a line search. We call this the (approximate) Riemannian conjugate gradient (RCG) algorithm.

Our RCG is an approximation of a true Riemannian conjugate gradient algorithm (Smith, 1993), in which the steps are taken along geodesic curves and tangent vectors evaluated at different points are transformed to the same tangent space using parallel transport along a geodesic. For small step sizes and geometries which are locally close to Euclidean, the approximations that we have made still retain many of the benefits of the exact algorithm while greatly simplifying the computations. Edelman et al. (1998) showed that near the solution Riemannian conjugate gradient method differs from the flat space version of conjugate gradient only by third order terms, and therefore both algorithms converge quadratically near the optimum. This convergence property is demonstrated in detail for our approximation in Appendix A.

The search direction for the RCG method is given by

$$\mathbf{p}_k = -\tilde{\mathbf{g}}_k + \beta \mathbf{p}_{k-1},$$

where $\tilde{\mathbf{g}}_k = \tilde{\nabla}\mathcal{F}(\boldsymbol{\xi})$ is the Riemannian gradient of Equation (4). The coefficient β is evaluated using the Polak-Ribiére formula (Nocedal, 1992; Smith, 1993; Edelman et al., 1998)

$$\beta = \frac{\langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1} \rangle_k}{||\tilde{\mathbf{g}}_{k-1}||^2_{k-1}}, \tag{5}$$

where $||\tilde{\mathbf{g}}_k||^2_k = \langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_k \rangle_k$ is the squared Riemannian norm of $\tilde{\mathbf{g}}_k$ in the tangent space where $\tilde{\mathbf{g}}_k$ is defined, and $\langle \mathbf{x}, \mathbf{y} \rangle_k$ denotes the Riemannian inner product of Equation (2) in the same tangent space.

We also apply a Riemannian version of the Powell-Beale restart method (Powell, 1977): the search direction is reset to the negative gradient direction if

$$|\langle \tilde{\mathbf{g}}_{k-1}, \tilde{\mathbf{g}}_k \rangle_k| \geq 0.2 ||\tilde{\mathbf{g}}_k||^2_k. \tag{6}$$

Compared to the traditional conjugate gradient, the Equations (5) and (6) are similar with just the dot products of the vectors replaced with Riemannian inner products.

Once the search direction is determined, we use standard line search to find the final update. Because the evaluation of the objective function is computationally costly, it is worthwhile to consider line search methods that stop earlier rather than wasting many function evaluations on fine tuning the parameters.

An example implementation of the algorithm is summarised in Algorithm 1. The inputs include the probabilistic model p, the form of used posterior approximation q, the initialisations for the variational parameters $\boldsymbol{\xi}$, and the data set $\boldsymbol{X}$ which is implicitly used in the objective function $\mathcal{F}$. The algorithm returns the variational parameters $\boldsymbol{\xi}$ that solve the learning and inference problem of $q(\boldsymbol{\theta}|\boldsymbol{\xi})$.

3.1 Computational Considerations

The RCG method is efficient as the geometry is defined by the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ and not the full model $p(\boldsymbol{X}|\boldsymbol{\theta})$ as in typical natural gradient methods. If the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is chosen such that disjoint groups of variables are independent, that is,

$$q(\boldsymbol{\theta}|\boldsymbol{\xi}) = \prod_i q_i(\boldsymbol{\theta}_i|\boldsymbol{\xi}_i),$$

Algorithm 1 An outline of an example Riemannian conjugate gradient algorithm for fixed-form VB. The presented method of integrating VB EM updates is only one of many possible alternatives.

function VB-RCG($p, q, \boldsymbol{\xi}_0 = (\boldsymbol{\xi}_0^{\mathrm{EM}}, \boldsymbol{\xi}_0^{\mathrm{RCG}}), \boldsymbol{X}$)
  $\mathbf{p}_0 = \mathbf{0}, \tilde{\mathbf{g}}_0 = \mathbf{1}$
  for $k = 1, 2, \ldots$ **do** ▷ Repeat until convergence
    for $\boldsymbol{\xi}^{(i)} \in \boldsymbol{\xi}^{\mathrm{EM}} = (\boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(N)})$ **do**
      $\boldsymbol{\xi}_k^{(i)} \leftarrow \arg\min_{\boldsymbol{\xi}^{(i)}} \mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}_k^{(1)}, \ldots, \boldsymbol{\xi}_k^{(i-1)}, \boldsymbol{\xi}^{(i)}, \boldsymbol{\xi}_{k-1}^{(i+1)}, \ldots, \boldsymbol{\xi}_{k-1}^{(N)}, \boldsymbol{\xi}_{k-1}^{\mathrm{RCG}}))$
      ▷ VB EM for some parameters
    $\tilde{\mathbf{g}}_k \leftarrow \mathbf{G}^{-1}(\boldsymbol{\xi}_{k-1}^{\mathrm{RCG}}) \nabla_{\boldsymbol{\xi}_{k-1}^{\mathrm{RCG}}} \mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}_k^{\mathrm{EM}}, \boldsymbol{\xi}_{k-1}^{\mathrm{RCG}}))$ ▷ Riemannian gradient
    $\beta \leftarrow \frac{\langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1} \rangle_k}{||\tilde{\mathbf{g}}_{k-1}||_{k-1}^2}$ ▷ Polak-Ribiére formula
    $\mathbf{p}_k \leftarrow -\tilde{\mathbf{g}}_k + \beta \mathbf{p}_{k-1}$ ▷ Update direction
    $\alpha \leftarrow \arg\min_\alpha \mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi}_k^{\mathrm{EM}}, \boldsymbol{\xi}_{k-1}^{\mathrm{RCG}} + \alpha \mathbf{p}_k))$ ▷ Line search
    $\boldsymbol{\xi}_k^{\mathrm{RCG}} \leftarrow \boldsymbol{\xi}_{k-1}^{\mathrm{RCG}} + \alpha \mathbf{p}_k$

the computation of the Riemannian gradient is simplified as the Fisher information matrix becomes block-diagonal. The required matrix operations can be performed very efficiently because

$$\mathrm{diag}(\mathbf{A}_1, \ldots, \mathbf{A}_n)^{-1} = \mathrm{diag}(\mathbf{A}_1^{-1}, \ldots, \mathbf{A}_n^{-1}).$$

The dimensionality of the problem space is often so high that working with the full matrix would not be feasible.

All vector operations needed in the RCG algorithm are of the form

$$\langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_l \rangle_m = \langle \mathbf{G}_k^{-1} \mathbf{g}_k, \mathbf{G}_l^{-1} \mathbf{g}_l \rangle_m = \mathbf{g}_k^T \mathbf{G}_k^{-T} \mathbf{G}_m \mathbf{G}_l^{-1} \mathbf{g}_l \tag{7}$$

for some iterate indices k, l, m. This is further simplified in case $m = k$ where $\mathbf{G}_k^{-T} \mathbf{G}_m = \mathbf{I}$ and in case $m = l$ where $\mathbf{G}_m \mathbf{G}_l^{-1} = \mathbf{I}$. Depending on the structure of the Fisher information matrix, the operations can be performed as a series of solving linear systems and matrix products to exploit the sparsity. A practical example of this in the case of a Gaussian approximation is presented in Section 3.2.1.

Finally, it is worth to note that when updating only a subset of variational parameters $\boldsymbol{\xi}$ at a time, many terms in $\mathcal{F}$ are constant and can be disregarded when finding a minimum.

3.2 Gaussian Approximation

Most obvious applications of the Riemannian gradient method are with a Gaussian approximation. In that case, it is most convenient to use a simple fixed-point update rule for the covariance and a Riemannian conjugate gradient update only for the mean.

Let us consider the optimisation of the free energy (1) when the approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is a multivariate Gaussian. The free energy can be decomposed as

$$\mathcal{F}(q(\boldsymbol{\theta}|\boldsymbol{\xi})) = E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \left\{ \log \frac{q(\boldsymbol{\theta}|\boldsymbol{\xi})}{p(\boldsymbol{X}, \boldsymbol{\theta})} \right\} = E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \left\{ \log q(\boldsymbol{\theta}|\boldsymbol{\xi}) \right\} + E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \left\{ -\log p(\boldsymbol{X}, \boldsymbol{\theta}) \right\}.$$

The former term is the negative entropy of the approximation, which in the case of a multivariate Gaussian $q(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is

$$E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left\{\log q(\boldsymbol{\theta}|\boldsymbol{\xi})\right\} = -\frac{1}{2}\log\det(2\pi e\boldsymbol{\Sigma}).$$

Straightforward differentiation yields a fixed point update rule for the covariance (Lappalainen and Miskin, 2000; Opper and Archambeau, 2009):

$$\boldsymbol{\Sigma}^{-1} = -2\nabla_{\boldsymbol{\Sigma}} E_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left\{\log p(\boldsymbol{X},\boldsymbol{\theta})\right\}, \tag{8}$$

where $\nabla_{\boldsymbol{\Sigma}}$ denotes the gradient with respect to $\boldsymbol{\Sigma}$. If the covariance matrix is assumed (block) diagonal, the same update rule applies for the (block) diagonal terms.

3.2.1 COMPUTING THE RIEMANNIAN METRIC TENSOR

For the univariate Gaussian distribution parametrised by mean and variance $q(\theta|\mu,v) = \mathcal{N}(\theta|\mu,v)$, we have

$$\log q(\theta|\mu,v) = -\frac{1}{2v}(\theta-\mu)^2 - \frac{1}{2}\log(v) - \frac{1}{2}\log(2\pi).$$

Furthermore,

$$E\left\{-\frac{\partial^2 \log q(\theta|\mu,v)}{\partial\mu^2}\right\} = \frac{1}{v}, \tag{9}$$

$$E\left\{-\frac{\partial^2 \log q(\theta|\mu,v)}{\partial v\partial\mu}\right\} = 0, \text{ and} \tag{10}$$

$$E\left\{-\frac{\partial^2 \log q(\theta|\mu,v)}{\partial v^2}\right\} = \frac{1}{2v^2}. \tag{11}$$

The vanishing of the cross term between the mean and the variance further supports using the simpler fixed point rule (8) to update the variances.

(a) (b) (c) (d)

Figure 3: The absolute change in the mean of the Gaussian in figures (a) and (b) and the absolute change in the variance of the Gaussian in figures (c) and (d) is the same. However, the relative effect is much larger when the variance is small as in figures (a) and (c) compared to the case when the variance is high as in figures (b) and (d) (Valpola, 2000).

In the case of univariate Gaussian distribution, the Riemannian gradient has a rather straightforward intuitive interpretation, which is illustrated in Figure 3. Compared to conventional gradient, Riemannian gradient compensates for the fact that changing the parameters of a Gaussian with small variance has much more pronounced effects than when the variance is large.

In case of multivariate Gaussian distribution parametrised by mean and covariance $q(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})$, the elements of the Fisher information matrix corresponding to the mean are simply

$$E\left\{-\frac{\partial^2 \log q(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}}\right\} = \boldsymbol{\Sigma}^{-1}. \tag{12}$$

The Fisher information matrix is thus equal to the precision matrix $\mathbf{G}_k = \boldsymbol{\Lambda}_k = \boldsymbol{\Sigma}_k^{-1}$.

Typically the covariance matrix $\boldsymbol{\Sigma}$ is assumed to have a simple structure that makes working with it very efficient. Possible structures for a covariance matrix include full, diagonal, block diagonal, a Gaussian Markov random field with a specific structure, and a factor analysis covariance $\boldsymbol{\Sigma} = \mathbf{D} + \sum_{i=1}^{k} \mathbf{v}\mathbf{v}^T$, where $\mathbf{D}$ is a diagonal matrix, or $\boldsymbol{\Sigma}^{-1} = \mathbf{K}^{-1} + \mathrm{diag}(\mathbf{v})$ with a fixed $\mathbf{K}$ and only N parameters in $\mathbf{v}$ for an N-variate Gaussian (Opper and Archambeau, 2009). It is also possible to derive the geometry for the covariance of a multivariate Gaussian. The result does, however, depend on the specific structure of the covariance.

Assuming a structured Gaussian Markov random field approximation with a tree or blocked tree structure, the precision matrix will be sparse with a simple structure. This allows efficient computation of the operations needed in Equation (7). $\boldsymbol{\Lambda}_l^{-1}\mathbf{g}_l$ (and correspondingly for k) can be computed by solving the linear system $\boldsymbol{\Lambda}_l\mathbf{x} = \mathbf{g}_l$, which can be done in $O(N)$ time for N variables using a propagation algorithm in the tree. For a blocked tree formed of N/n blocks of size n, the complexity is $O(n^2N)$. Examples of such algorithms for chains are given in Golub and Loan (1996), but a general tree can be handled analogously. The complexity of multiplication by $\boldsymbol{\Lambda}$ is similar.

Examples of Gaussian Markov random fields with this structure can be easily found in time series models, where the approximation for the state sequence $\boldsymbol{S} = (\mathbf{s}(1),\ldots,\mathbf{s}(T))$ is typically either a single "blocked" chain

$$q(\boldsymbol{S}) = \prod_{t=2}^{T} q(\mathbf{s}(t)|\mathbf{s}(t-1))q(\mathbf{s}(1))$$

or a product of independent chains

$$q(\boldsymbol{S}) = \prod_i \left[q(s_i(1)) \prod_{t=2}^{T} q(s_i(t)|s_i(t-1)) \right]. \tag{13}$$

In a d dimensional model of length T, the time complexity of the Riemannian vector operations in RCG is $O(d^3T)$ for the former and $O(dT)$ for the latter.

4. Case Study: Mixture of Gaussians

As the first case study, we consider the mixture-of-Gaussians (MoG) model as was done by Kuusela et al. (2009). In this case, we applied the RCG also for variables with a non-Gaussian approximation. Furthermore, the conjugate-exponential nature of the MoG model allows direct comparison with VB EM.

4.1 The Mixture-of-Gaussians Model

We consider a finite mixture of K Gaussians (Attias, 2000; Bishop, 2006)

$$p(\mathbf{x}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k),$$

The model	$p(\mathbf{X}\|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Lambda}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\mathcal{N}(\mathbf{x}_n\|\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k^{-1})^{z_{nk}}$ $p(\mathbf{Z}\|\boldsymbol{\pi}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\pi_k^{z_{nk}}$
Key variables	$\mathbf{Z} = (z_{nk}), \boldsymbol{\theta} = (\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k, \boldsymbol{\pi})$
The approximation	$q(\mathbf{Z},\boldsymbol{\theta}) = q(\mathbf{Z})q(\boldsymbol{\theta})$
The update algorithm	Joint RCG updates for $\mathbf{Z}$ and means of $\boldsymbol{\mu}_k$, fixed point updates for variances of $\boldsymbol{\mu}_k$, VB EM updates for the rest

Table 1: Summary of the mixture-of-Gaussians model

where $\mathbf{x}$ is a D-dimensional random vector, $\boldsymbol{\pi} = [\pi_1 \cdots \pi_K]^T$ are the mixing coefficients, and $\boldsymbol{\mu}_k$ and $\boldsymbol{\Lambda}_k$ are the mean vector and the precision matrix of the kth Gaussian component, respectively.

In the case of the MoG model, the binary latent variables $\mathbf{Z}$ denote which one of the K Gaussian components has generated a particular observation $\mathbf{x}_n$ with $z_{nk} = 1$ denoting the component responsible for generating the observed data point $\mathbf{x}_n$. Let N denote the total number of observed data points.

Given the mixing coefficients $\boldsymbol{\pi}$, the probability distribution over the latent variables is given by

$$p(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\pi_k^{z_{nk}}.$$

The mixing coefficients have a conjugate Dirichlet prior

$$p(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_0),$$

where $\boldsymbol{\alpha}_0 = [\alpha_0, \ldots, \alpha_0]^T$.

Similarly, the likelihood can be written as

$$p(\mathbf{X}|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Lambda}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k^{-1})^{z_{nk}}.$$

In this case, the conjugate prior for the component parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ is given by the Gaussian-Wishart distribution

$$p(\boldsymbol{\mu},\boldsymbol{\Lambda}) = p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}) = \prod_{k=1}^{K}\mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_0,(\beta_0\boldsymbol{\Lambda}_k)^{-1})\mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_0,\nu_0),$$

where the Wishart distribution is defined up to a normalising constant by the equation

$$\mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W},\nu) \propto |\boldsymbol{\Lambda}|^{(\nu-D-1)/2}\exp\left(-\frac{1}{2}\mathrm{Tr}(\mathbf{W}^{-1}\boldsymbol{\Lambda})\right).$$

The joint distribution over all the random variables of the model is then given by

$$p(\mathbf{X},\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) = p(\mathbf{X}|\mathbf{Z},\boldsymbol{\mu},\boldsymbol{\Lambda})p(\mathbf{Z}|\boldsymbol{\pi})p(\boldsymbol{\pi})p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}).$$

This resulting model can be illustrated with the graphical model shown in Figure 4.

We now make the factorising approximation

$$q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) = q(\mathbf{Z})q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}),$$

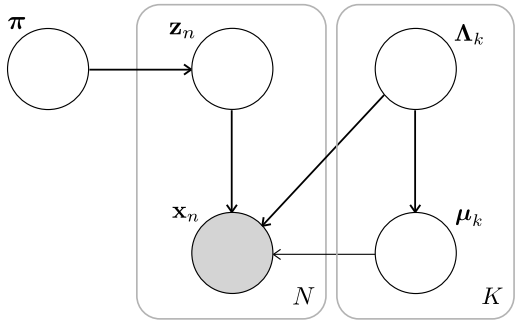


Figure 4: A graphical model representing the MoG model (Attias, 2000; Bishop, 2006), where the hyperparameters have been omitted for clarity. The observed data $\mathbf{X}$ are marked with a shaded circle. The rectangular plates denote the repetition of N observations $\mathbf{x}_n$ along with corresponding latent variables $\mathbf{z}_n$, and of the parameters of K mixture components.

which leads to an update rule for $q(\mathbf{Z})$ (VB-E step) and subsequently an update rule for $q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})$ (VB-M step). The resulting approximate posterior distributions are

$$q(\mathbf{Z}) = \prod_{n=1}^{N}\prod_{k=1}^{K} r_{nk}^{z_{nk}} \tag{14}$$

and

$$q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) = q(\boldsymbol{\pi})q(\boldsymbol{\mu},\boldsymbol{\Lambda}) = q(\boldsymbol{\pi})\prod_{k=1}^{K} q(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k),$$

where

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \tag{15}$$

and

$$q(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k,(\beta_k\boldsymbol{\Lambda}_k)^{-1})\,\mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_k,\nu_k). \tag{16}$$

The derivations of the VB EM and RCG learning algorithms for the MoG model are presented in Appendix B.

4.2 Experiments

The hyperparameters are set to the following values for all the experiments: $\alpha_0 = 1$, $\beta_0 = 1$, $\nu_0 = D$, $\mathbf{W}_0 = \frac{4}{D}\mathbf{I}$ and $\mathbf{m}_0 = \mathbf{0}$. These values can be interpreted to describe our prior beliefs of the model when we anticipate having Gaussian components near the origin but are fairly uncertain about the number of the components.

The maximum number of components is set to $K = 8$ unless otherwise mentioned with each component having a randomly generated initial mean $\mathbf{m}_k$ drawn from a Gaussian distribution with zero mean and covariance $0.16\mathbf{I}$. Other distribution parameters are initially set to the following values: $\alpha_k = 1$, $\beta_k = 10$, $\nu_k = D$ and $\mathbf{W}_k = \frac{4}{D}\mathbf{I}$ for all k. The Powell-Beale restart scheme of Equation (6) is not used. Instead, the search direction is reset to the negative gradient direction

after $\sqrt{n}$ iterations, where n is the number of parameters updated with the gradient method. The optimisation is assumed to have converged when the improvement in free energy $|\mathcal{F}^t - \mathcal{F}^{t-1}| < \varepsilon$ for two consecutive iterations with ε being separately specified for each of the experiments below.

It should be noted that this convergence criterion favours methods such as VB EM which typically takes more of cheaper and smaller steps, while the Riemannian gradient algorithm takes fewer larger steps that are computationally more demanding and take longer to reach the preset improvement threshold.

Because different initial means can produce significantly different results in terms of the required CPU time and achieved final free energy, all the experiments are repeated 30 times with different initialisations.

The artificial data set used to compare the different algorithms in learning the MoG model was drawn from a mixture of 5 spherical two-dimensional Gaussians with equal weights. The mean of the first component is at the origin while the means of the others are $(\pm R, \pm R)$. The constant $R = 0.3$ unless otherwise mentioned and the covariance of all the components is $0.03\mathbf{I}$.

When different gradient-based algorithms are compared using the artificial data containing $N = 500$ data points, the results shown in Figure 5 are obtained. It can be seen that the standard gradient descent and conjugate gradient (CG) algorithms have problems locating even a decent optimum within a reasonable time. Clearly the convergence criterion, which was set to $\varepsilon = 10^{-7}N$, is too lax for these algorithms as the simulations are terminated before convergence to a good solution. Using the Riemannian gradient (RG) and further RCG radically improves the performance. Based on the curves, even the standard CG algorithm is more than 10 times slower than RCG. This experiment was conducted using a fairly small number of observations, a lax convergence criterion and the maximum number of components $K = 5$ in order to allow the standard gradient to finish in a reasonable time.

We also considered the L-BFGS algorithm (Byrd et al., 1995; Carbonetto, 2007) as a higher order Euclidean algorithm. L-BFGS is a limited-memory version of the popular quasi-Newton BFGS algorithm. It can be seen as a compromise between the fast converging but memory-intensive quasi-Newton methods and the less efficient conjugate gradient methods better suited for medium- to large-scale problems. The degree of this compromise is controlled by the memory length parameter m which was set to $m = 20$ in Figure 5. It was found out that, regardless of the value of m, the performance of L-BFGS was very similar to CG with the small deviations explained by the differences in the line search methods employed by the algorithms. It also turned out that the line search subroutine of the L-BFGS implementation had a tendency to converge prematurely to a poor solution. In order to circumvent this, the convergence criterion of the L-BFGS had to be tightened by a factor of 10^{-9} compared to the gradient-based algorithms.

We next compare RCG, RG and VB EM using different values of R. The number of observations was increased to $N = 1000$ and the convergence criterion was set to $\varepsilon = 10^{-12}N$ in order to maximise the quality of the optima found. Figure 6 shows the median CPU time required for convergence for the different algorithms in 30 repeated experiments. It can be seen that with small values of R, RCG outperforms VB EM, while with large values of R, VB EM performs slightly better. This means that, at least in terms of this experiment, RCG performs better than VB EM when the latent variables are difficult to infer from the data.

Given the discussion of Section 2.3, it is not surprising to note that both VB EM and RG perform qualitatively in a fairly similar manner in the experiment of Figure 6. It should especially be noted that both methods suffer from significant slowdowns near the values of $R = 0.2$ and $R = 0.325$. On

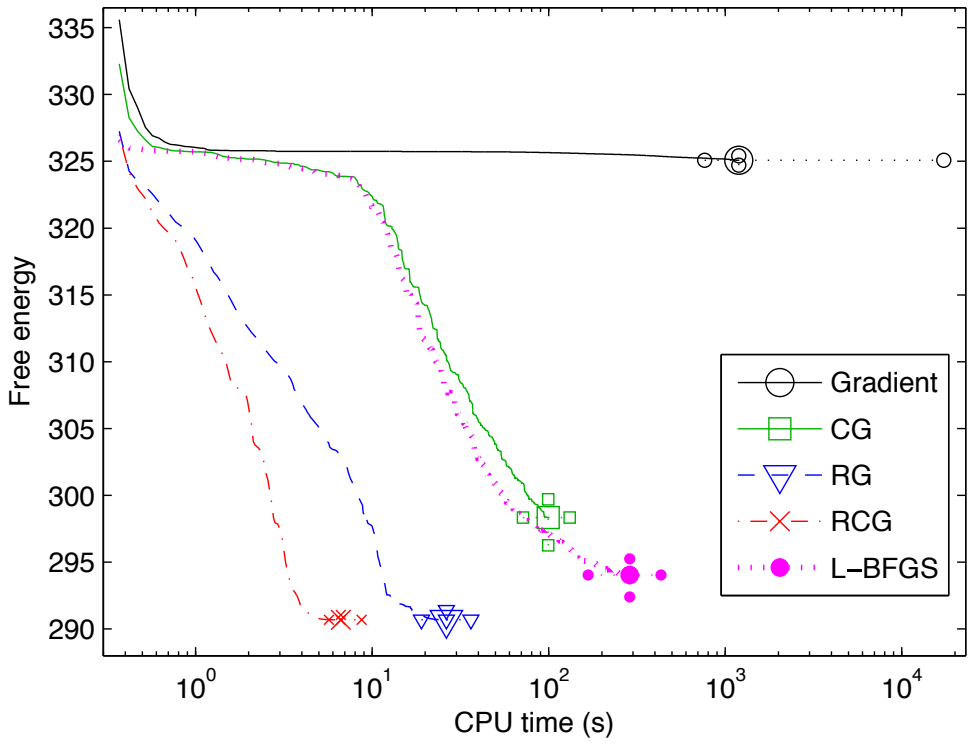


Figure 5: Convergence curves of gradient-based algorithms using the MoG model for artificial data with $R = 0.3$. The algorithms compared are the standard gradient descent, the conjugate gradient (CG), the Riemannian gradient (RG) and the Riemannian conjugate gradient (RCG) methods as well as the limited-memory BFGS (L-BFGS) algorithm. The curves shown are medians of 30 simulations drawn up to the median termination time. The smaller marks denote 25% and 75% quantiles of the termination time in the horizontal direction and the corresponding quantiles of the free energy at the median termination time in the vertical direction. Note that the time scale is logarithmic.

the other hand, the use of conjugate directions in the Riemannian space seems to result in a fairly uniform performance across all values of R.

There is some variation in the quality of the optima the different methods converge to. This is illustrated in Figure 7 for RCG and VB EM. There is no evidence for either algorithm consistently producing better results than the other. The only outlier in this data is with $R = 0.225$ where VB EM is the only algorithm to discover the global optimum. This is by no means typical, and with other data sets we have seen RCG sometimes consistently finding better optima. Based on the figure, RCG seems slightly more sensitive to local optima, but the result is not qualitatively different from VB EM which, to some extent, also suffers from the same problem.

Although the time complexity of one step of each of the compared algorithms is linear in the number of samples, the algorithms nevertheless perform differently as the number of samples in-

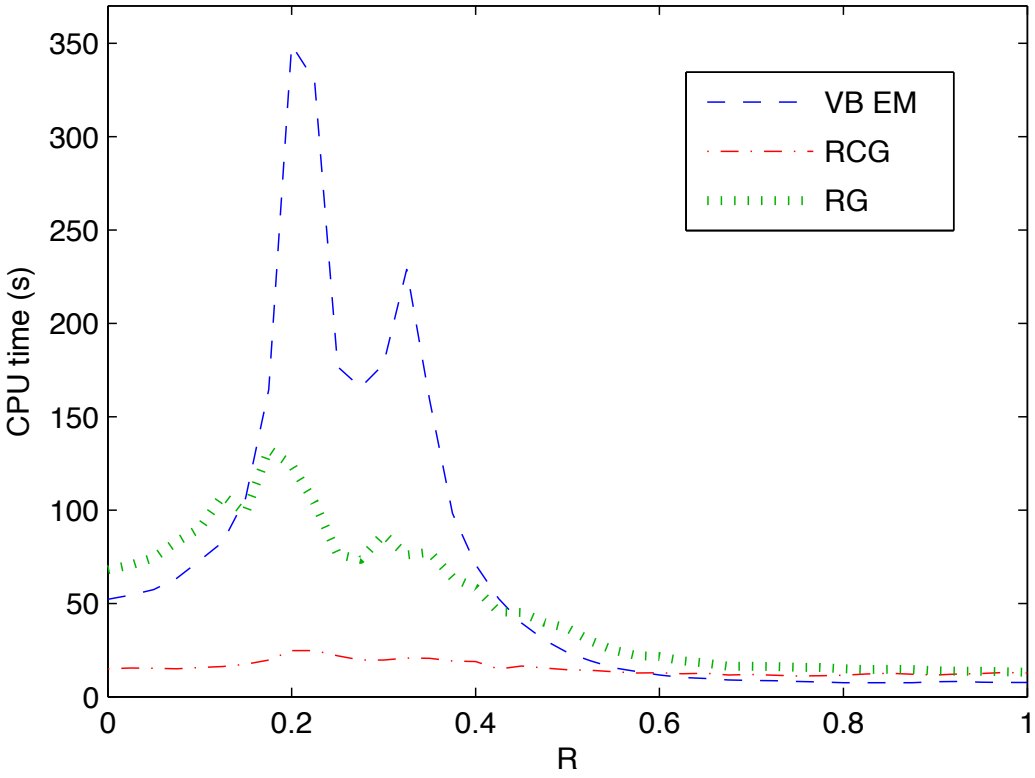


Figure 6: Comparison of VB EM with the Riemannian gradient (RG) and the Riemannian conjugate gradient (RCG) methods using the MoG model with the artificial data. The curves show the median CPU time required for convergence as a function of R. VB EM slows down significantly at the critical overlap of $R \approx 0.2$, while RCG is almost unaffected. A similar slowdown also affects RG implying that the use of conjugate directions contributes to the nearly uniform running time of RCG.

creases. To see this, we set $\varepsilon = 10^{-8}N$ and probed a wide range of values of N. The results are illustrated in Figure 8 which shows that VB EM slows down faster than linearly as the number of samples increases. The most likely reason for this behaviour is that the posterior will be more peaked when the number of observations is large and this slows down the alternating VB EM iteration. The same phenomenon also affects RCG, but the effect is much stronger in VB EM. The results suggest that especially for large data sets, it can be worthwhile to consider alternatives to basic batch VB EM, such as on-line algorithms (Sato, 2001) or gradient-based methods.

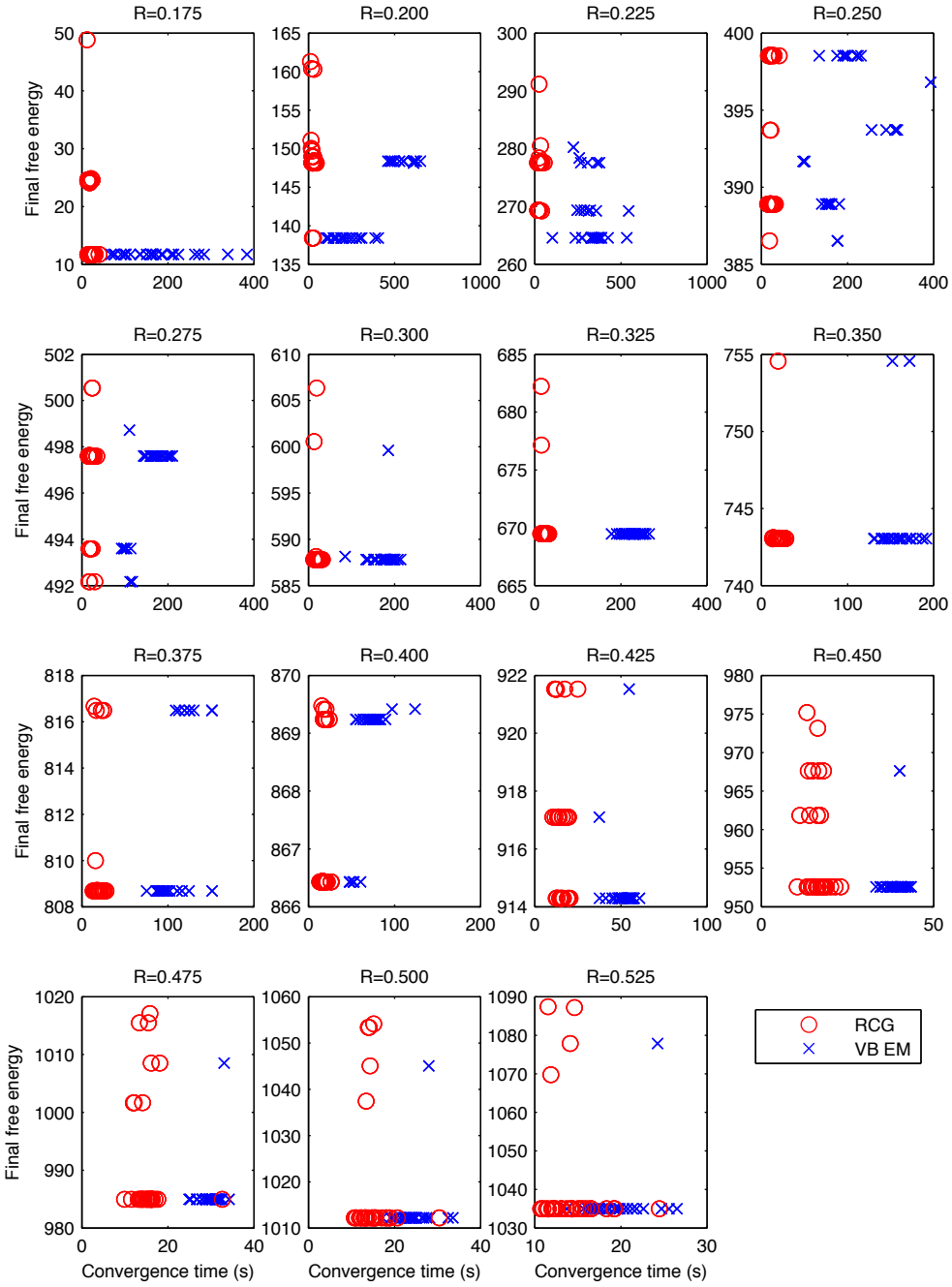


Figure 7: Final free energy value as a function of running time in the critical parameter range in the MoG experiment with varying R. Both RCG and VB EM sometimes fail to converge to the global optimum. Interestingly, there usually is no correlation between the quality of the solution and convergence time.

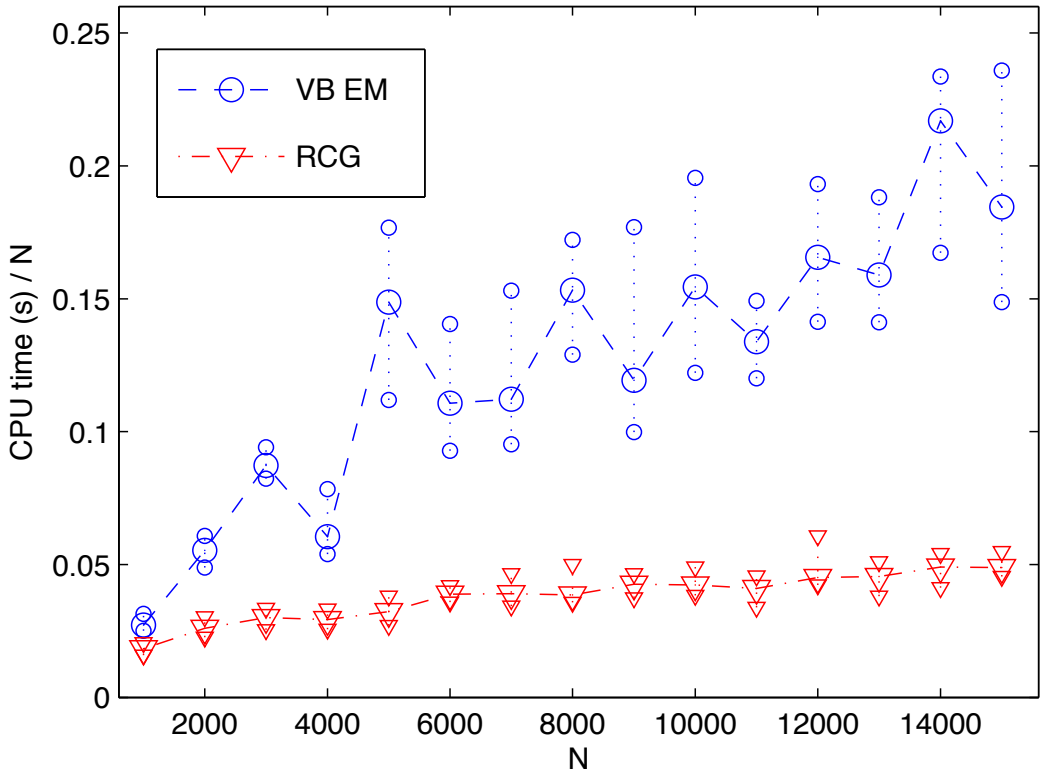


Figure 8: Median convergence times of 30 simulations of the MoG model on artificial data as a function of the number of observations N with VB EM and the Riemannian conjugate gradient (RCG) algorithms. The smaller marks indicate 25% and 75% quantiles.

The model	$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t)$ $\mathbf{s}(t) = \mathbf{s}(t-1) + \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) + \mathbf{m}(t)$
Key variables	$\boldsymbol{\theta}_S = (\mathbf{s}(t))$, $\boldsymbol{\theta}_\boldsymbol{\theta} = (\boldsymbol{\theta}_\mathbf{f}, \boldsymbol{\theta}_\mathbf{g}, \mathbf{v}_m, \mathbf{v}_n)$
The approximation	$q(\boldsymbol{\theta}_S, \boldsymbol{\theta}_\boldsymbol{\theta}) = q(\boldsymbol{\theta}_S) q(\boldsymbol{\theta}_\boldsymbol{\theta})$, where $q(\boldsymbol{\theta}_\boldsymbol{\theta})$ is Gaussian with diagonal covariance and $q(\boldsymbol{\theta}_S)$ Gaussian with tridiagonal precision (see text)
The update algorithm	Joint RCG updates for the means of $\boldsymbol{\theta}_S, \boldsymbol{\theta}_\mathbf{f}, \boldsymbol{\theta}_\mathbf{g}$, fixed point updates for their covariances, VB EM updates for the rest

Table 2: Summary of the nonlinear state-space model

5. Case Study: Nonlinear State-Space Models

As the second case study, we consider the nonlinear state-space model (NSSM) introduced by Valpola and Karhunen (2002). The model is specified by the generative model

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t), \tag{17}$$

$$\mathbf{s}(t) = \mathbf{s}(t-1) + \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) + \mathbf{m}(t), \tag{18}$$

where t is time, $\mathbf{x}(t)$ are the observations, and $\mathbf{s}(t)$ are the hidden states. The observation mapping $\mathbf{f}$ and the dynamical mapping $\mathbf{g}$ are nonlinear and they are modelled with multi-layer perceptron (MLP) networks whose weight matrices are included in $\boldsymbol{\theta}_\mathbf{f}$ and $\boldsymbol{\theta}_\mathbf{g}$. The observation noise vector $\mathbf{n}$ and process noise vector $\mathbf{m}$ are assumed Gaussian with zero mean and covariances $\mathrm{diag}(\exp(2\mathbf{v}_n))$ and $\mathrm{diag}(\exp(2\mathbf{v}_m))$. The latent states $\mathbf{s}(t)$ are commonly denoted by $\boldsymbol{\theta}_{\boldsymbol{S}}$. The model parameters include both the weights of the MLP networks and a number of hyperparameters. The posterior approximation of these parameters is a Gaussian with a diagonal covariance matrix. The posterior approximation of the states $q(\boldsymbol{\theta}_{\boldsymbol{S}}|\boldsymbol{\xi}_{\boldsymbol{S}})$ is a Gaussian Markov random field with a correlation between the corresponding components of subsequent state vectors $s_j(t)$ and $s_j(t-1)$, as in Equation (13). This is a realistic minimum assumption for modelling the dependence of the state vectors $\mathbf{s}(t)$ and $\mathbf{s}(t-1)$ (Valpola and Karhunen, 2002). Omitted details of the model are presented in Appendix C and a summary is given in Table 2.

Because of the nonlinearities the model is not in the conjugate exponential family, and the standard VB learning methods are only applicable to hyperparameters but not to the latent states or weights of the MLPs. The free energy (1) can nevertheless be evaluated by linearising the MLP networks $\mathbf{f}$ and $\mathbf{g}$ (Honkela and Valpola, 2005; Honkela et al., 2007). This allows evaluating the gradient with respect to $\boldsymbol{\xi}_{\boldsymbol{S}}$, $\boldsymbol{\xi}_\mathbf{f}$, and $\boldsymbol{\xi}_\mathbf{g}$ and using a gradient based optimiser to adapt the parameters. Combining Equations (4) and (12), the Riemannian gradient for the mean elements is given by

$$\tilde{\nabla}_{\boldsymbol{\mu}_q} \mathcal{F}(\boldsymbol{\xi}) = \boldsymbol{\Sigma}_q \nabla_{\boldsymbol{\mu}_q} \mathcal{F}(\boldsymbol{\xi}),$$

where $\boldsymbol{\mu}_q$ is the mean of the variational approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ and $\boldsymbol{\Sigma}_q$ is the corresponding covariance. The covariance matrix of the model parameters is diagonal while the inverse covariance matrix of the latent states $\mathbf{s}(t)$ is block-diagonal with tridiagonal blocks. This implies that all computations with these can be done in linear time with respect to the number of the parameters. The covariances are updated separately using a fixed-point update rule similar to (8) as described by Valpola and Karhunen (2002). A complete derivation of the free energy of the model is presented in Appendix C.

5.1 Experiments

We applied the method for learning nonlinear state-space models presented above to real world speech data. Experiments were conducted with different data sizes to study the performance differences between the algorithms. The data consisted of 21 dimensional mel-frequency log power speech spectra of continuous human speech. This is a detailed representation of speech signals similar to those often used in speech recognition. A segment of 100 samples corresponds to approximately 0.8 seconds of speech. The task is to learn a nonlinear dynamical model for this data.

To study the performance differences between the Riemannian conjugate gradient (RCG) method, standard Riemannian gradient (RG) method, standard conjugate gradient (CG) method and the heuristic algorithm from Valpola and Karhunen (2002), the algorithms were applied to different sized parts of the speech data set. Unfortunately a reasonable comparison with a VB EM algorithm was impossible because an extended-Kalman-filter-based VB EM algorithm failed with the nonlinear model.

The size of the data subsets varied between 200 and 500 samples. A five-dimensional state-space was used. The MLP networks for the observation and dynamical mappings had 20 hidden nodes. Four different initialisations and two different segments of data of each size were used, resulting in eight repetitions for each algorithm and data size. The results for different data segments of the

same size were pooled together as the convergence times were in general very similar. An algorithm was assumed to have converged when $|\mathcal{F}^t - \mathcal{F}^{t-1}| < \varepsilon = (10^{-5}N/80)$ for 5 consecutive iterations, where $\mathcal{F}^t$ is the free energy at iteration t and N is the size of the data set. Alternatively, the iteration was stopped after 24 hours even if it had not converged.

The MLP network is notoriously prone to local optima. Practically all our simulations converged to local optima with different parameter estimates, but there were no statistically significant differences in the free energies corresponding to these optima attained by different algorithms (Wilcoxon rank-sum test, 5 % significance level). In practice, the free energy values tend to have a very strong correlation with predictive performance of the model (Honkela et al., 2007). There were still some differences, and especially the RG algorithm with smaller data sizes often appeared to converge very early to an extremely poor solution. These were filtered by removing results where the attained free energy that was more than two RCG standard deviations worse than RCG average for the particular data set. Thus all the used results are from runs converging to a roughly equally good solution. The results of one run where the heuristic algorithm diverged were also discarded from the analysis.

The results can be seen in Figure 9. The plain CG and RG methods were clearly slower than others and the maximum runtime was reached by most CG and some RG runs. RCG was clearly the fastest algorithm with the heuristic method of Valpola and Karhunen (2002) between these extremes. The observed differences are, save for a few exceptions mostly with smaller data sets, statistically significant (Wilcoxon rank-sum test, 5 % significance level).

As a more realistic example, a larger data set of 1000 samples was used to train a seven-dimensional state-space model. In this experiment both MLP networks of the NSSM had 30 hidden nodes. The convergence criterion was $\varepsilon = 10^{-6}$ and the maximum runtime was 72 hours. The performances of the RCG, RG, CG methods and the heuristic algorithm were compared. The results can be seen in Figure 10. The results show the convergence for five different initialisations with markers at the end showing when the convergence was reached. It should be noted that the scale of the CPU time axis is logarithmic.

RCG clearly outperformed the other algorithms in this experiment as well. In particular, both RG and CG hit the maximum runtime in every run, and especially CG was nowhere near convergence at this time. RCG also outperformed the heuristic algorithm (Valpola and Karhunen, 2002) by a factor of more than 10.

6. Case Study: Nonlinear Factor Analysis

The model	$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t)$ $\mathbf{s}(t) = \mathcal{N}(0, \mathrm{diag}(\exp(2\mathbf{v}_s)))$
Key variables	$\boldsymbol{S} = (\mathbf{s}(t)), \boldsymbol{\theta} = (\boldsymbol{\theta}_\mathbf{f}, \mathbf{v}_s, \mathbf{v}_n)$
The approximation	$q(\boldsymbol{S}, \boldsymbol{\theta}) = q(\boldsymbol{S})q(\boldsymbol{\theta})$, where both $q(\boldsymbol{\theta})$ and $q(\boldsymbol{S})$ are Gaussian with diagonal covariances
The update algorithm	Joint RCG updates for means of $\boldsymbol{S}, \boldsymbol{\theta}_\mathbf{f}$, fixed point update for their covariances, VB EM updates for the rest

Table 3: Summary of the nonlinear factor analysis model

As the final case study, the RCG and RG methods were implemented as extensions to the VB nonlinear factor analysis (NFA) method (Lappalainen and Honkela, 2000; Honkela and Valpola,

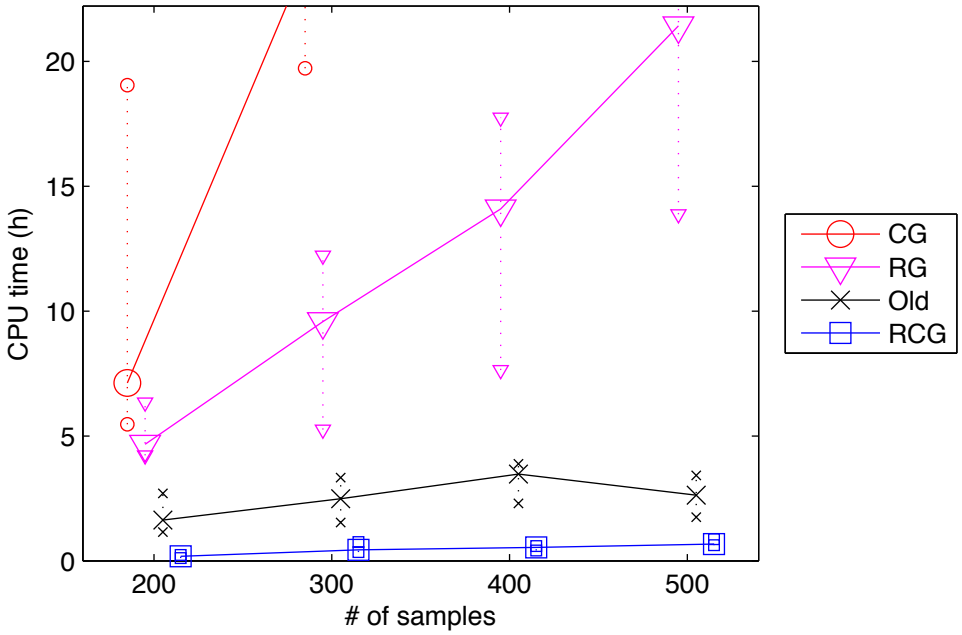


Figure 9: Convergence speed of the Riemannian conjugate gradient (RCG), the Riemannian gradient (RG) and the conjugate gradient (CG) methods as well as the heuristic algorithm (Old) with different data sizes of the speech data set and the nonlinear state-space model. The lines show median times with 25 % and 75 % quantiles shown by the smaller marks. The times were limited to at most 24 hours, which was reached by a number of simulations.

2005; Honkela et al., 2007). NFA models the mapping between latent factors $\mathbf{s}(t)$ and observations $\mathbf{x}(t)$ with an MLP as in Equation (17):

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t).$$

Instead of the dynamical model of Equation (18), $\mathbf{s}(t)$ has independent Gaussian priors with a unit covariance. As NFA can be seen as a special case of NSSM with no dynamic mapping, the implementation is straightforward. The model is summarised in Table 3. Complete derivation of the model and a learning algorithm based on conjugate gradients is presented by Honkela et al. (2007). The generalisation for Riemannian gradient is straightforward as the Fisher information matrix is diagonal.

The RCG, RG and CG methods were applied for learning an NFA model for parts of the speech data set, a different part of which was also used in the NSSM experiments. As the NFA model cannot capture the dynamics of speech, the experiment aimed at finding a nonlinear embedding of speech on a lower dimensional manifold. To stimulate this, we drew a suitable subset of samples randomly from the full data set of 7860 samples, excluding any dynamical relations in the data. Silent segments were excluded from the data.

We tested each algorithm for data sets ranging in size from 300 to 1000 samples, running 10 simulations with different random initialisations for every setting. The results are shown in Figure

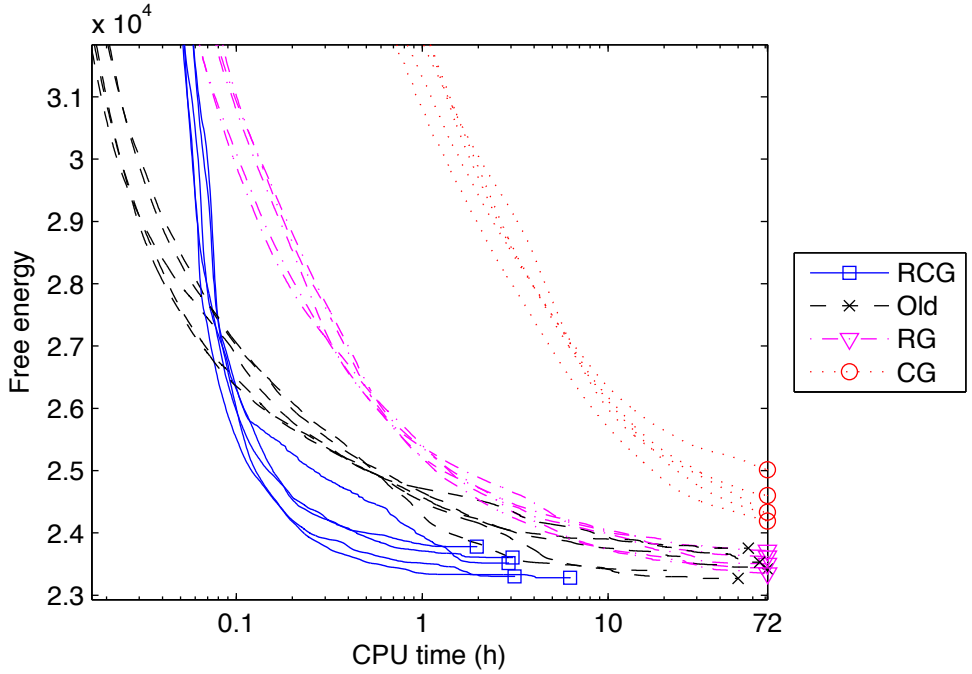


Figure 10: Comparison of the performance of the Riemannian conjugate gradient (RCG), the Riemannian gradient (RG), the conjugate gradient (CG) methods and the heuristic algorithm with the full speech data set of 1000 samples using the nonlinear state-space model. The free energy $\mathcal{F}$ is plotted against computation time using a logarithmic time scale. The tick marks show when simulations either converged or were terminated after 72 hours.

11. RCG is again clearly superior to both RG and CG, but CG is now faster than RG. The observed differences are statistically significant (Wilcoxon rank-sum test, 5 % significance level), except between CG and RG for 300 samples.

7. Discussion

The proposed RCG algorithm combines two improvements over plain gradient optimisation: use of Riemannian gradient and conjugate gradients. One interesting feature in the experimental results is the relative performance of the conjugate gradient and Riemannian gradient algorithms that implement only one of these. Conjugate gradient is faster than Riemannian gradient for NFA, but the opposite is true for NSSM and MoG. Especially the latter differences are quite significant and consistent across several different data sets. One obvious difference between the models is that for NFA the Fisher information matrix is diagonal while for NSSM and MoG this is not the case. This suggests that the Riemannian gradient approach may be the most useful when the metric is more complex, although more careful analysis would be needed to properly understand the effects of different improvements.

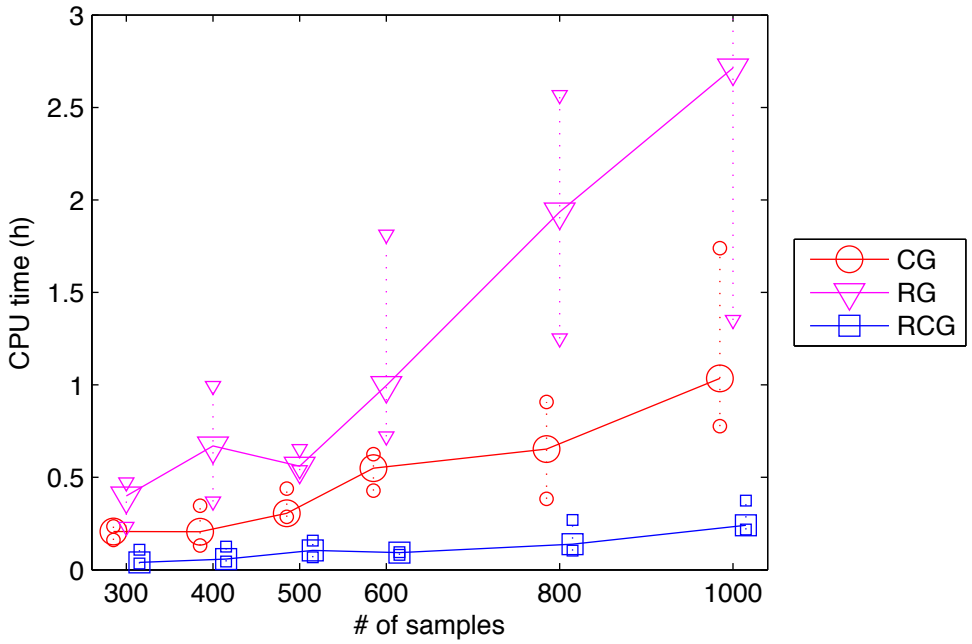


Figure 11: Convergence speed of the Riemannian conjugate gradient (RCG), the Riemannian gradient (RG) and the conjugate gradient (CG) methods with different data sizes of the speech dimensionality reduction data set with the nonlinear factor analysis model. The lines show median times with 25 % and 75 % quantiles shown by the smaller marks.

As illustrated by the MoG example, the RCG algorithm can also be applied to conjugate-exponential models to replace the more common VB EM algorithm. In practice, simpler and more straightforward EM acceleration methods based on, for example, pattern search or adaptive over-relaxation (see, e.g., Honkela et al., 2003; Salakhutdinov and Roweis, 2003) may still provide comparable or better results with less human effort. These methods are only applicable when EM itself is applicable, though.

The experiments in this paper show that using even a greatly simplified variant of the Riemannian conjugate gradient method for some variables is enough to acquire a large speedup. Considering univariate Gaussian distributions, the regular gradient is prone to overemphasise changes to model variables with small posterior variance and underemphasise variables with large posterior variance, as seen from Equations (9)–(11). The posterior variance of latent variables is often much larger than the posterior variance of model parameters and the Riemannian gradient takes this into account in a very natural manner.

The Riemannian conjugate gradient method differs from Euclidean superlinear optimisation methods such as quasi-Newton methods in that it uses higher-order information of the geometry of the parameter space, but not of the function being optimised. These are essentially two independent avenues for improvement: it would be possible, although complicated, to derive a Riemannian quasi-Newton method. Our experiments clearly show that in these problems, a proper model of the

geometry appears significantly more important than using higher-order information of the objective function.

In this paper, we have presented a Riemannian conjugate gradient learning algorithm for fixed-form variational Bayes. The RCG algorithm provides an efficient method for VB learning in models that do not belong to the conjugate-exponential family as required by the standard variational EM algorithm. For suitably structured approximations, the computational overhead from using Riemannian gradients instead of conventional gradients is negligible. In practical examples, the Riemannian gradient approach provided several orders of magnitude speedups over conventional gradient algorithms, thus making VB learning of these models practical on a much larger scale.

MATLAB code for all the models used in the case studies is available at `http://www.cis.hut.fi/projects/bayes/software/ncg/`.

Acknowledgments

This work was supported in part by the Academy of Finland under its Centres of Excellence in Research Program, and the IST Program of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. AH and TR were supported by Postdoctoral Researchers' projects of the Academy of Finland (No 121179, 133145). TR was also supported by the Academy of Finland project "Unsupervised machine learning in latent variable models" (No 121802). This publication only reflects the authors' views.

Appendix A. Convergence of the Riemannian Conjugate Gradient Algorithm

The Riemannian conjugate gradient algorithm has similar superlinear convergence properties to the Euclidean space conjugate gradient algorithm. Assuming the objective $\mathcal{F}(\boldsymbol{\xi})$ has continuous third order derivatives and that there exist $m > 0, M$ such that the Hessian $H(\boldsymbol{\xi})$ satisfies

$$m\mathbf{x}^T\mathbf{x} \leq \mathbf{x}^T H(\boldsymbol{\xi})\mathbf{x} \leq M\mathbf{x}^T\mathbf{x}$$

for all $\boldsymbol{\xi}$ and $\mathbf{x}$, the error decreases quadratically over N steps in an N-dimensional problem. Thus, denoting the optimum by $\boldsymbol{\alpha}$ and the iterates by $\boldsymbol{\xi}_i$, we have (Edelman et al., 1998; Cohen, 1972)

$$||\boldsymbol{\alpha} - \boldsymbol{\xi}_{i+N}|| \leq C||\boldsymbol{\alpha} - \boldsymbol{\xi}_i||^2 \tag{19}$$

in some neighbourhood of $\boldsymbol{\alpha}$.

We now show that the approximations in the RCG algorithm, namely ignoring the parallel transports and performing line searches along straight lines instead of geodesics, do not effect the convergence rate of Equation (19).

Theorem 1 *Assuming the objective $\mathcal{F}(\boldsymbol{\xi})$ has bounded derivatives for up to third order and that the Fisher information $\mathbf{G}(\boldsymbol{\xi})$ is smooth in a neighbourhood of the solution, the RCG algorithm performing a line search along a straight line in the direction*

$$\mathbf{p}_k = -\tilde{\mathbf{g}}_k + \beta\mathbf{p}_{k-1},$$

where $\tilde{\mathbf{g}}_k$ is the Riemannian gradient and

$$\beta = \frac{\langle \tilde{\mathbf{g}}_k, \tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1}\rangle_k}{||\tilde{\mathbf{g}}_{k-1}||^2_{k-1}}$$

to find the sequence of iterates $\boldsymbol{\xi}_k$ shares the same convergence property of Equation (19) in some ε-neighbourhood of the solution as the Riemannian conjugate gradient algorithm performing a line search along a geodesic in the direction

$$\mathbf{p}_k^* = -\tilde{\mathbf{g}}_k^* + \beta^* \boldsymbol{\tau} \mathbf{p}_{k-1}^*,$$

where $\tilde{\mathbf{g}}_k$ is the Riemannian gradient, $\boldsymbol{\tau}\mathbf{p}_{k-1}^*$ is the vector $\mathbf{p}_{k-1}^*$ after parallel transport to the starting point of the new search and

$$\beta^* = \frac{\langle \tilde{\mathbf{g}}_k^*, \tilde{\mathbf{g}}_k^* - \boldsymbol{\tau}\tilde{\mathbf{g}}_{k-1}^* \rangle_{k^*}}{||\tilde{\mathbf{g}}_{k-1}^*||^2_{(k-1)^*}},$$

where $\langle \cdot, \cdot \rangle_{k^*}$ is the inner product at $\boldsymbol{\xi}_k^*$, to find the sequence of iterates $\boldsymbol{\xi}_k^*$.

Proof Let us assume that the two algorithms are started at $\boldsymbol{\xi}_k = \boldsymbol{\xi}_k^*$ and $||\boldsymbol{\xi}_k - \boldsymbol{\alpha}|| < \varepsilon$ for some $\varepsilon > 0$. We show by induction on i that $\mathbf{p}_{i-1} = \mathbf{p}_{i-1}^* + O(\varepsilon^2)$ and $\boldsymbol{\xi}_i = \boldsymbol{\xi}_i^* + O(\varepsilon^2)$ for $i \geq k$ which is sufficient to prove the theorem.

The base case is trivial as $\boldsymbol{\xi}_k = \boldsymbol{\xi}_k^*$ and $\mathbf{p}_{k-1} = \mathbf{p}_{k-1}^* = 0$ at the start of the algorithm.

Assume now that the claim is valid for $i = k, \ldots, K$, and let us prove it for $i = K+1$. From Edelman et al. (1998) we know that

$$\begin{aligned} \boldsymbol{\xi}(\varepsilon) &= \boldsymbol{\xi}(0) + \varepsilon \Delta / ||\Delta|| + O(\varepsilon^3), \\ \boldsymbol{\tau}\tilde{\mathbf{g}}(\varepsilon) &= \tilde{\mathbf{g}} + O(\varepsilon^2), \end{aligned}$$

where $\boldsymbol{\xi}(\varepsilon)$ is a geodesic in direction Δ, $\boldsymbol{\tau}\tilde{\mathbf{g}}(\varepsilon)$ is the parallel transport of $\tilde{\mathbf{g}}$ to $\boldsymbol{\xi}(\varepsilon)$. The norms of $\tilde{\mathbf{g}}$ and $\mathbf{p}$ are also of the order $O(\varepsilon)$.

The assumption $\boldsymbol{\xi}_K = \boldsymbol{\xi}_K^* + O(\varepsilon^2)$ implies that the gradients evaluated at these points satisfy $\tilde{\mathbf{g}}_K = \tilde{\mathbf{g}}_K^* + O(\varepsilon^2)$. Furthermore,

$$\langle \mathbf{x}, \mathbf{y} \rangle_{K^*} = \mathbf{x}^T \mathbf{G}(\boldsymbol{\xi}_K^*) \mathbf{y} = \mathbf{x}^T (\mathbf{G}(\boldsymbol{\xi}_K) + O(\varepsilon^2)) \mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle_K + O(\varepsilon^2 ||\mathbf{x}||_K ||\mathbf{y}||_K).$$

Using the above asymptotics,

$$\begin{aligned} \beta^* &= \frac{\langle \tilde{\mathbf{g}}_K^*, \tilde{\mathbf{g}}_K^* - \boldsymbol{\tau}\tilde{\mathbf{g}}_{K-1}^* \rangle_{K^*}}{||\tilde{\mathbf{g}}_{K-1}^*||^2_{(K-1)^*}} = \frac{\langle \tilde{\mathbf{g}}_K + O(\varepsilon^2), \tilde{\mathbf{g}}_K - \tilde{\mathbf{g}}_{K-1} + O(\varepsilon^2) \rangle_{K^*}}{||\tilde{\mathbf{g}}_{K-1} + O(\varepsilon^2)||^2_{(K-1)^*}} \\ &= (1 + O(\varepsilon)) \frac{\langle \tilde{\mathbf{g}}_K, \tilde{\mathbf{g}}_K - \tilde{\mathbf{g}}_{K-1} \rangle_{K^*} + O(\varepsilon^3)}{||\tilde{\mathbf{g}}_{K-1}||^2_{(K-1)^*}} = (1 + O(\varepsilon)) \frac{\langle \tilde{\mathbf{g}}_K, \tilde{\mathbf{g}}_K - \tilde{\mathbf{g}}_{K-1} \rangle_K + O(\varepsilon^3)}{||\tilde{\mathbf{g}}_{K-1}||^2_{(K-1)} + O(\varepsilon^3)} \\ &= (1 + O(\varepsilon)) \left(\frac{\langle \tilde{\mathbf{g}}_K, \tilde{\mathbf{g}}_K - \tilde{\mathbf{g}}_{K-1} \rangle_K}{||\tilde{\mathbf{g}}_{K-1}||^2_{(K-1)}} + O(\varepsilon) \right) = \beta + O(\varepsilon). \end{aligned}$$

Similarly we can find the difference in the search direction,

$$\begin{aligned} \mathbf{p}_K^* &= -\tilde{\mathbf{g}}_K^* + \beta^* \boldsymbol{\tau} \mathbf{p}_{K-1}^* = -\tilde{\mathbf{g}}_K + \beta^* \mathbf{p}_{K-1}^* + O(\varepsilon^2) = -\tilde{\mathbf{g}}_K + \beta \mathbf{p}_{K-1} + O(\varepsilon^2) \\ &= \mathbf{p}_K + O(\varepsilon^2), \end{aligned}$$

which completes the first part of the induction step.

The corresponding step lengths, t^* and t, also differ by $O(\varepsilon)$. To show this, let us use the Taylor expansion of f about the optimum $\boldsymbol{\alpha}$:

$$\begin{aligned}\mathcal{F}(\boldsymbol{\xi}) &= \mathcal{F}(\boldsymbol{\alpha}) + \frac{1}{2}(\boldsymbol{\xi}-\boldsymbol{\alpha})^T H(\boldsymbol{\alpha})(\boldsymbol{\xi}-\boldsymbol{\alpha}) + O(\varepsilon^3),\\ \nabla\mathcal{F}(\boldsymbol{\xi}) &= H(\boldsymbol{\alpha})(\boldsymbol{\xi}-\boldsymbol{\alpha}) + O(\varepsilon^2).\end{aligned}$$

The line search finds the zero of $\mathbf{p}_K \cdot \nabla\mathcal{F}(\boldsymbol{\xi})$ along the line $\boldsymbol{\xi} = \boldsymbol{\xi}_{K-1} + t\mathbf{p}_K$, which yields

$$\mathbf{p}_K^T H(\boldsymbol{\alpha}) \left[(\boldsymbol{\xi}_{K-1} + t\mathbf{p}_K - \boldsymbol{\alpha}) + O(\varepsilon^2)\right] = 0,$$

which can be solved to obtain

$$t = -\frac{\mathbf{p}_K^T H(\boldsymbol{\alpha})(\boldsymbol{\xi}_{K-1}-\boldsymbol{\alpha})}{\mathbf{p}_K^T H(\boldsymbol{\alpha})\mathbf{p}_K} + O(\varepsilon),$$

where we have used the fact that $\mathbf{p}_K^T H(\boldsymbol{\alpha})\mathbf{p}_K \geq m||\mathbf{p}_K||^2$.

Correspondingly, the exact algorithm finds the zero along the geodesic from $\boldsymbol{\xi}^*_{K-1}$ in the direction $\mathbf{p}^*_K$ of $(\tau\mathbf{p}^*_K)^T\nabla\mathcal{F}(\boldsymbol{\xi})$

$$\begin{aligned}&(\tau\mathbf{p}^*_K(t^*||\mathbf{p}^*_K||))^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}^*_{K-1}(t^*||\mathbf{p}^*_K||) - \boldsymbol{\alpha} + O(\varepsilon^2)] =\\ &\qquad (\mathbf{p}^*_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}^*_{K-1} + t^*\mathbf{p}^*_K - \boldsymbol{\alpha} + O(\varepsilon^2)] + O(\varepsilon^3) = 0,\end{aligned}$$

where $\boldsymbol{\xi}_{K-1}(t^*||\mathbf{p}^*_K||)$ is the geodesic starting from $\boldsymbol{\xi}^*_{K-1}$ in the direction $\mathbf{p}^*_K$, and $\tau\mathbf{p}^*_K(t^*||\mathbf{p}^*_K||)$ is the parallel transport of $\mathbf{p}^*_K$ along this geodesic. The solution of this equation yields

$$\begin{aligned}t^* &= -\frac{(\mathbf{p}^*_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}^*_{K-1} - \boldsymbol{\alpha} + O(\varepsilon^3)]}{(\mathbf{p}^*_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})\mathbf{p}^*_K} + O(\varepsilon)\\ &= -\frac{(\mathbf{p}_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}_{K-1} - \boldsymbol{\alpha} + O(\varepsilon^2)]}{(\mathbf{p}_K + O(\varepsilon^2))^T H(\boldsymbol{\alpha})[\mathbf{p}_K + O(\varepsilon^2)]} + O(\varepsilon)\\ &= -(1+O(\varepsilon))\frac{\mathbf{p}_K^T H(\boldsymbol{\alpha})[\boldsymbol{\xi}_{K-1} - \boldsymbol{\alpha}] + O(\varepsilon^3)}{\mathbf{p}_K^T H(\boldsymbol{\alpha})\mathbf{p}_K} + O(\varepsilon) = t + O(\varepsilon).\end{aligned}$$

Now

$$\begin{aligned}\boldsymbol{\xi}^*_K &= \boldsymbol{\xi}^*_{K-1}(t^*||\mathbf{p}^*_K||) = \boldsymbol{\xi}^*_{K-1} + t^*\mathbf{p}^*_K + O(\varepsilon^3)\\ &= \boldsymbol{\xi}_{K-1} + O(\varepsilon^2) + [t + O(\varepsilon)][\mathbf{p}_K + O(\varepsilon^2)] + O(\varepsilon^3) = \boldsymbol{\xi}_K + O(\varepsilon^2),\end{aligned}$$

which completes the proof. ■

Appendix B. Derivations of the Mixture-of-Gaussians Model

In this section we present details of the variational MoG model, including necessary EM updates, the free energy and the metric tensor for the RCG algorithm.

B.1 VB EM for the Mixture-of-Gaussians Model

This section is completely based on the variational treatment of the MoG model of Attias (2000) and Bishop (2006). Because of this, some details of the derivation of the VB EM algorithm for the MoG model will be omitted here and we will concentrate only on the most important results.

In expressing the update rules for the distribution parameters in Equations (14), (15) and (16), we will find the following definitions useful:

$$
\begin{aligned}
N_k &= \sum_{n=1}^{N} r_{nk}, \\
\bar{\mathbf{x}}_k &= \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n, \\
\mathbf{S}_k &= \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \bar{\mathbf{x}}_k)(\mathbf{x}_n - \bar{\mathbf{x}}_k)^T, \\
\log \tilde{\Lambda}_k &= \sum_{i=1}^{D} \psi\left(\frac{\nu_k + 1 - i}{2} \right) + D \log 2 + \log |\mathbf{W}_k|, \\
\log \tilde{\pi}_k &= \psi(\alpha_k) - \psi\left(\sum_{k'=1}^{K} \alpha_{k'} \right),
\end{aligned}
$$

where D is the dimensionality of the data and $\psi(\cdot)$ is the digamma function which is defined as the derivative of the logarithmic gamma function, that is

$$
\psi(x) = \frac{d}{dx} \log \Gamma(x).
$$

Using these definitions, the parameters r_{nk} of the approximate posterior over latent variables $q(\mathbf{Z})$ which are updated in the E-step are given by

$$
r_{nk} = \frac{\rho_{nk}}{\sum_{l=1}^{K} \rho_{nl}},
$$

where

$$
\rho_{nk} = \tilde{\pi}_k \tilde{\Lambda}_k^{1/2} \exp\left(-\frac{D}{2\beta_k} - \frac{\nu_k}{2} (\mathbf{x}_n - \mathbf{m}_k)^T \mathbf{W}_k (\mathbf{x}_n - \mathbf{m}_k) \right).
$$

The parameters r_{nk} are called *responsibilities* because they represent the responsibility the kth component takes in explaining the nth observation. The responsibilities can be arranged into a matrix $\mathbf{R} = (r_{nk})$ and will have to satisfy the following conditions

$$
0 \leq r_{nk} \leq 1, \tag{20}
$$

$$
\sum_{k=1}^{K} r_{nk} = 1. \tag{21}
$$

The parameter update equations for the M-step are then given by

$$\alpha_k = \alpha_0 + N_k, \tag{22}$$
$$\beta_k = \beta_0 + N_k, \tag{23}$$
$$\nu_k = \nu_0 + N_k + 1, \tag{24}$$
$$\mathbf{m}_k = \frac{1}{\beta_0 + N_k}(\beta_0 \mathbf{m}_0 + N_k \bar{\mathbf{x}}_k),$$
$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\bar{\mathbf{x}}_k - \mathbf{m}_0)(\bar{\mathbf{x}}_k - \mathbf{m}_0)^T. \tag{25}$$

B.2 The Free Energy

The free energy of Equation (1) is

$$\begin{aligned}
\mathcal{F} &= \sum_{\mathbf{Z}} \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\Lambda}} q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \log \frac{q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})}{p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})} d\boldsymbol{\pi} d\boldsymbol{\mu} d\boldsymbol{\Lambda} \\
&= E_q\{\log q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\} - E_q\{\log p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\} \\
&= E_q\{\log q(\mathbf{Z})\} + E_q\{\log q(\boldsymbol{\pi})\} + E_q\{\log q(\boldsymbol{\mu}, \boldsymbol{\Lambda})\} \\
&\quad - E_q\{\log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\} - E_q\{\log p(\mathbf{Z}|\boldsymbol{\pi})\} \\
&\quad - E_q\{\log p(\boldsymbol{\pi})\} - E_q\{\log p(\boldsymbol{\mu}, \boldsymbol{\Lambda})\}.
\end{aligned} \tag{26}$$

These expectations can be evaluated to give (Bishop, 2006)

$$E_q\{\log q(\mathbf{Z})\} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \log r_{nk},$$
$$E_q\{\log q(\boldsymbol{\pi})\} = \sum_{k=1}^{K} (\alpha_k - 1) \log \tilde{\pi}_k + \log C(\boldsymbol{\alpha}),$$
$$E_q\{\log q(\boldsymbol{\mu}, \boldsymbol{\Lambda})\} = \sum_{k=1}^{K} \left\{ \frac{1}{2} \log \tilde{\Lambda}_k + \frac{D}{2} \log \frac{\beta_k}{2\pi} - \frac{D}{2} - H_q\{\boldsymbol{\Lambda}_k\} \right\},$$
$$E_q\{\log p(\mathbf{Z}|\boldsymbol{\pi})\} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \log \tilde{\pi}_k,$$
$$E_q\{\log p(\boldsymbol{\pi})\} = \log C(\boldsymbol{\alpha}_0) + (\alpha_0 - 1) \sum_{k=1}^{K} \log \tilde{\pi}_k,$$

$$E_q\{\log p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\} = \frac{1}{2} \sum_{k=1}^{K} N_k \left\{ \log \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) - \nu_k (\bar{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\bar{\mathbf{x}}_k - \mathbf{m}_k) - D \log 2\pi \right\},$$

$$E_q\{\log p(\boldsymbol{\mu}, \boldsymbol{\Lambda})\} = \frac{1}{2} \sum_{k=1}^{K} \left\{ D \log \frac{\beta_0}{2\pi} + \log \tilde{\Lambda}_k - \frac{D\beta_0}{\beta_k} - \beta_0 \nu_k (\mathbf{m}_k - \mathbf{m}_0)^T \mathbf{W}_k (\mathbf{m}_k - \mathbf{m}_0) \right\}$$
$$+ K \log B(\mathbf{W}_0, \nu_0) + \frac{\nu_0 - D - 1}{2} \sum_{k=1}^{K} \log \tilde{\Lambda}_k - \frac{1}{2} \sum_{k=1}^{K} \nu_k \operatorname{Tr}(\mathbf{W}_0^{-1} \mathbf{W}_k),$$

where $\mathrm{Tr}(\mathbf{A})$ denotes the trace of matrix $\mathbf{A}$ and $H_q\{\mathbf{\Lambda}_k\}$ is the entropy of the distribution $q(\mathbf{\Lambda}_k)$. The functions C and B are defined by the following two equations:

$$C(\boldsymbol{\alpha}) = \Gamma\left(\sum_{k=1}^{K}\alpha_k\right)\left(\prod_{k=1}^{K}\Gamma(\alpha_k)\right)^{-1},$$

$$B(\mathbf{W},\nu) = |\mathbf{W}|^{-\nu/2}\left(2^{\nu D/2}\pi^{D(D-1)/4}\prod_{i=1}^{D}\Gamma\left(\frac{\nu+1-i}{2}\right)\right)^{-1}.$$

B.3 Riemannian Conjugate Gradient for the Mixture-of-Gaussians Model

To be able to compare the VB EM and RCG algorithms, we assume that the approximate posterior distribution $q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\mathbf{\Lambda})$ takes the same functional form as in the case of the VB EM algorithm. Thus, the fixed form posterior distributions are given by Equations (14), (15) and (16) and the free energy which is to be minimised by the RCG algorithm is given Equation (26). In this work, we will only be optimising the responsibilities r_{nk} and the means $\mathbf{m}_k$ using gradient-based methods. All other model parameters, namely the parameters α_k of the Dirichlet distribution, the parameters β_k controlling the covariance of the component means as well as the parameters $\mathbf{W}_k$ and ν_k of the Wishart distribution, are updated using the VB EM update Equations (22), (23), (24) and (25).

There are a few things that have to be taken into account when deriving gradient-based algorithms for the MoG model. Firstly, the responsibilities have to satisfy the constraints given by Equations (20) and (21). This can be enforced by using the *softmax* parametrisation

$$r_{nk} = \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}}. \tag{27}$$

It can be easily seen that by using this parametrisation the responsibilities are always positive and $\sum_{k=1}^{K} r_{nk} = 1$. As a results it holds that $0 \leq r_{nk} \leq 1$ and we can conduct unconstrained optimisation in the $\boldsymbol{\gamma}$ space.

Secondly, if we set the responsibilities $r_{nk}, n = 1\ldots N, k = 1\ldots K-1$ to some values, the values of $r_{nK}, n = 1\ldots N$ are given by condition (21), that is $r_{nK} = 1 - \sum_{k=1}^{K-1} r_{nk}$. As a result, the number of degrees of freedom in the responsibilities of the model is not the number of responsibilities NK but instead $N(K-1)$. When we are using the parametrisation (27), this means that we can regard the parameters γ_{nK} as constants and only optimise the free energy with respect to parameters $\gamma_{nk}, n = 1\ldots N, k = 1\ldots K-1$. This is especially important when using the Riemannian gradient.

The gradient of the free energy (26) with respect to $\mathbf{m}_k$ is given by

$$\nabla_{\mathbf{m}_k}\mathcal{F} = \nu_k\mathbf{W}_k(N_k(\mathbf{m}_k - \bar{\mathbf{x}}_k) + \beta_0(\mathbf{m}_k - \mathbf{m}_0))$$

and the derivative with respect to γ_{nk} is given by

$$\frac{\partial\mathcal{F}}{\partial\gamma_{nk}} = E_{nk} - r_{nk}F_n,$$

where

$$E_{nk} = r_{nk}\left(\log r_{nk} - \log\tilde{\pi}_k - \frac{1}{2}\left(\log\tilde{\Lambda}_k - \frac{D}{\beta_k} - D\log 2\pi - \nu_k(\mathbf{x}_n - \mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_n - \mathbf{m}_k)\right)\right)$$

and

$$F_n = \sum_{k=1}^{K} E_{nk}.$$

We can update the responsibilities r_{nk} without having to evaluate and store the parameters γ_{nk} by noting that

$$\begin{aligned} r'_{nk} &= \frac{e^{\gamma_{nk}+\Delta\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}+\Delta\gamma_{nl}}} \\ &= \frac{\sum_{l=1}^{K} e^{\gamma_{nl}}}{\sum_{l=1}^{K} e^{\gamma_{nl}+\Delta\gamma_{nl}}} \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}} e^{\Delta\gamma_{nk}} = c_n r_{nk} e^{\Delta\gamma_{nk}}, \end{aligned}$$

where r'_{nk} is the new responsibility, $\Delta\gamma_{nk}$ is the change in parameter γ_{nk} determined by a line search in the search direction and c_n is a normalising constant which makes sure that $\sum_{k=1}^{K} r'_{nk} = 1$. Thus c_n can also be expressed in the form $c_n = (\sum_{k=1}^{K} r_{nk} e^{\Delta\gamma_{nk}})^{-1}$ and we can update the responsibilities using the formula

$$r'_{nk} = \frac{r_{nk} e^{\Delta\gamma_{nk}}}{\sum_{l=1}^{K} r_{nl} e^{\Delta\gamma_{nl}}}.$$

In order to use the Riemannian gradient, we need to know the Riemannian metric tensor $\mathbf{G}$ of the parameter space $(\mathbf{m},\boldsymbol{\gamma})$ which is given by Equation (3). The resulting matrix is a block diagonal matrix with blocks $\mathbf{A}_k = \beta_k \nu_k \mathbf{W}_k$ for each $\mathbf{m}_k$ and blocks $\mathbf{B}_n = -\mathbf{r}_n^T \mathbf{r}_n + \mathrm{diag}(\mathbf{r}_n)$ for each sample, where $\mathbf{r}_n$ is the nth row of the responsibility matrix $\mathbf{R}$ except for element r_{nK}, that is $\mathbf{r}_n = [r_{n1} \cdots r_{nK-1}]$. $\mathrm{diag}(\mathbf{a})$ is used here to denote a square matrix which has the elements of vector $\mathbf{a}$ on its main diagonal. This block-diagonal structure of the matrix makes the Riemannian vector operations easy and efficient to implement.

Since the EM updates of parameters are computationally efficient compared to the evaluation of the objective function, it is more efficient to do them also within the line search of the RCG update rather than as a separate step as in Algorithm 1.

Appendix C. Derivation of the Nonlinear State-Space Model

In this section we review the details of the nonlinear state-space model of Valpola and Karhunen (2002).

C.1 Probability Model and Priors

The nonlinear state-space model of Valpola and Karhunen (2002) can be described with these two equations:

$$\mathbf{s}(t) \sim \mathcal{N}(\mathbf{s}(t-1) + \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}), \mathrm{diag}(\exp(2\mathbf{v}_m))), \tag{28}$$

$$\mathbf{x}(t) \sim \mathcal{N}(\mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}), \mathrm{diag}(\exp(2\mathbf{v}_n))), \tag{29}$$

where $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ denotes a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. The nonlinear mappings are modelled with MLP networks:

$$\begin{aligned} \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) &= \mathbf{D}\tanh[\mathbf{C}\mathbf{s}(t-1) + \mathbf{c}] + \mathbf{d}, \\ \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) &= \mathbf{B}\tanh[\mathbf{A}\mathbf{s}(t) + \mathbf{a}] + \mathbf{b}. \end{aligned}$$

The model for data $\mathbf{X}$ is thus described using unobserved variables

$$\boldsymbol{\theta} = (\mathbf{s}(t), \mathbf{A}, \mathbf{a}, \mathbf{B}, \mathbf{b}, \mathbf{C}, \mathbf{c}, \mathbf{D}, \mathbf{d}, \mathbf{v}_m, \mathbf{v}_n).$$

The priors of the variables are specified to fix the scaling ambiguity between $\mathbf{s}$ and $\mathbf{A}$ and to have a hierarchical prior allowing automatic relevance determination (ARD) (Bishop, 2006) like decisions to inactivate parts of the model:

$$\begin{aligned}
A_{ij} &\sim \mathcal{N}(0,1),\\
\Phi_{ij} &\sim \mathcal{N}(0, \exp(2v_{\Phi_j})),\\
\phi_i &\sim \mathcal{N}(m_\phi, \exp(2v_\phi)),\\
v_{\nu_i} &\sim \mathcal{N}(m_{v_\nu}, \exp(2v_{v_\nu})),\\
v_{\Phi_j} &\sim \mathcal{N}(m_{v_\Phi}, \exp(2v_{v_\Phi})),
\end{aligned}$$

where $\nu \in \{m,n\}$, $\phi \in \{a,b,c,d\}$ and $\Phi \in \{B,C,D\}$. All the hyperparameters have vague priors $\mathcal{N}(0, 100^2)$.

C.2 Posterior Approximation

In order to allow efficient learning, the posterior approximation $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{-1})$ is restricted to be Gaussian with mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and precision (inverse covariance) $\boldsymbol{\Lambda}_{\boldsymbol{\theta}}$. Furthermore, the precision of the approximation is restricted to be almost diagonal. The only allowed off-diagonal terms are in the approximation of $\mathbf{s}(t)$ which includes a correlation between $s_i(t)$ and $s_i(t+1)$. Different components of the state vector $\mathbf{s}(t)$ are still assumed independent, and the posterior approximation of the states is a product of independent chains.

Following the theory of Gaussian Markov random fields, this assumption translates to a tridiagonal precision (inverse covariance) matrix with non-zero elements only on the main diagonal and on the diagonal corresponding to the assumed links. The corresponding covariance matrix has full blocks for each component of the state.

C.3 The Free Energy

In order to derive the value of the free energy (1), we note that

$$\mathcal{F}(q(\boldsymbol{\theta})) = E_{q(\boldsymbol{\theta})}\{\log q(\boldsymbol{\theta})\} + E_{q(\boldsymbol{\theta})}\{-\log p(\mathbf{X}, \boldsymbol{\theta})\}.$$

The first term is the negative entropy of a Gaussian

$$E_{q(\boldsymbol{\theta})}\{\log q(\boldsymbol{\theta})\} = -\frac{N}{2}\log(2\pi e) - \frac{1}{2}\log\det\boldsymbol{\Lambda}_{\boldsymbol{\theta}},$$

where N is the dimensionality of $\boldsymbol{\theta}$. The second term splits to a sum of a number of terms according to Equations (28)–(29).

$$E_{q(\boldsymbol{\theta})}\{-\log p(\mathbf{X}, \boldsymbol{\theta})\} = \sum_{t,i} E_{q(\boldsymbol{\theta})}\{-\log p(x_i(t)|\boldsymbol{\theta})\} + \sum_{\gamma \in \boldsymbol{\theta}} E_{q(\boldsymbol{\theta})}\left\{-\log p(\gamma|\boldsymbol{\theta}_{\setminus\gamma})\right\},$$

where $\boldsymbol{\theta}_{\setminus\gamma}$ denotes the parameters γ depends on.

The terms in the sum are expectations for parameters γ following a normal model $\mathcal{N}(m, e^{2v})$. The negative logarithm of the pdf is

$$-\log p(\gamma|\boldsymbol{\theta}_{\setminus\gamma}) = \frac{1}{2}\log(2\pi) + v + \frac{1}{2}(\gamma - m)^2 \exp(-2v).$$

Assuming independent Gaussian approximations[2] for γ, m and v with means $\bar{\gamma}, \bar{m}, \bar{v}$ and variances $\tilde{\gamma}, \tilde{m}, \tilde{v}$, respectively, the expectation is

$$E_{q(\boldsymbol{\theta})}\left\{-\log p(\gamma|\boldsymbol{\theta}_{\setminus\gamma})\right\} = \frac{1}{2}\log(2\pi) + \bar{v} + \frac{1}{2}[(\bar{\gamma} - \bar{m})^2 + \tilde{\gamma} + \tilde{m}]\exp(2\tilde{v} - 2\bar{v}).$$

For the observations $x_i(t)$ we obtain similarly

$$E_{q(\boldsymbol{\theta})}\left\{-\log p(x_i(t)|\boldsymbol{\theta})\right\} = \frac{1}{2}\log(2\pi) + \bar{v}_{n_i} + \frac{1}{2}[(x - \bar{f}_i(t))^2 + \tilde{f}_i(t)]\exp(2\tilde{v}_{n_i} - 2\bar{v}_{n_i}),$$

where the means $\bar{f}_i(t)$ and variances $\tilde{f}_i(t)$ of $\mathbf{f}(\mathbf{s}(t))$ are evaluated as explained in Honkela and Valpola (2005); Honkela et al. (2007).

For the states $s_i(t)$ we can similarly derive (Valpola and Karhunen, 2002)

$$\begin{aligned} E_{q(\boldsymbol{\theta})}\left\{-\log p(s_i(t)|\boldsymbol{\theta}_{\setminus\mathbf{s}(t)})\right\} &= \frac{1}{2}\log(2\pi) + \bar{v}_{m_i} \\ &+ \frac{1}{2}\left[(\bar{s}_i(t) - \bar{g}_i(t))^2 + \tilde{s}_i(t) + \tilde{g}_i(t) - 2\check{s}_i(t, t-1)\frac{g_i(t)}{s_i(t-1)}\tilde{s}_i(t-1)\right]\exp(2\tilde{v}_{m_i} - 2\bar{v}_{m_i}), \end{aligned}$$

where $\bar{g}_i(t)$ and $\tilde{g}_i(t)$ are the mean and variance of $\mathbf{g}(\mathbf{s}(t-1))$ evaluated similarly as those of $\mathbf{f}(\mathbf{s}(t))$, and $\check{s}_i(t, t-1)$ is the linear correlation between $s_i(t-1)$ and $s_i(t)$ as explained in Valpola and Karhunen (2002). The partial derivative $\frac{g_i(t)}{s_i(t-1)}$ is evaluated naturally as a by-product of evaluation of $\tilde{g}_i(t)$ as explained previously (Honkela and Valpola, 2005; Honkela et al., 2007).

C.4 Update Rules

The hyperparameters $v_{\Phi_j}, m_\phi, v_\phi, v_{\nu_i}, m_{v_\nu}, v_{v_\nu}, v_{\Phi_j}, m_{v_\Phi}, v_{v_\Phi}$ are updated using a VB EM type scheme to find a global optimum, given current values of the other parameters (Lappalainen and Miskin, 2000). The variances of the states and the weights of the MLP networks are updated using the fixed-point rule and the means by the RCG algorithm, as described in Section 5.

References

S. Amari. *Differential-Geometrical Methods in Statistics*, volume 28 of *Lecture Notes in Statistics*. Springer-Verlag, Berlin, 1985.

S. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1995. doi: 10.1016/0893-6080(95)00003-8.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

2. We will use the notation $\bar{\gamma}$ for the mean and $\tilde{\gamma}$ for the variance of γ in the approximation for all variables.

S. Amari and H. Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, RI, USA, 2000.

C. Archambeau, M. Opper, Y. Shen, D. Cornford, and J. Shawe-Taylor. Variational inference for diffusion processes. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 17–24. MIT Press, Cambridge, MA, USA, 2008.

H. Attias. A variational Bayesian framework for graphical models. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 209–215. MIT Press, Cambridge, MA, USA, 2000.

D. Barber and C. Bishop. Ensemble learning for multi-layer networks. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 395–401. The MIT Press, Cambridge, MA, USA, 1998.

C. M. Bishop. *Pattern Recognition and Machince Learning*. Springer, New York, NY, USA, 2006.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. doi: 10.1137/0916069.

P. Carbonetto. A MATLAB interface for L-BFGS-B. `http://people.cs.ubc.ca/~pcarbo/lbfgsb-for-matlab.html`, March 2007.

A. I. Cohen. Rate of convergence of several conjugate gradient algorithms. *SIAM Journal of Numerical Analysis*, 9(2):248–259, 1972. doi: 10.1137/0709024.

A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998. doi: 10.1137/S0895479895290954.

Z. Ghahramani and M. Beal. Propagation algorithms for variational Bayesian learning. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 507–513. The MIT Press, Cambridge, MA, USA, 2001.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *J. of the Royal Statistical Society, Series B (Methodological)*, 2011. In press.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, 3rd edition, 1996.

A. González and J. R. Dorronsoro. Natural conjugate gradient training of multilayer perceptrons. *Neurocomputing*, 71(13–15):2499–2506, 2008. doi: 10.1016/j.neucom.2007.11.035.

A. Honkela and H. Valpola. Unsupervised variational Bayesian learning of nonlinear models. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 593–600. MIT Press, Cambridge, MA, USA, 2005.

A. Honkela, H. Valpola, and J. Karhunen. Accelerating cyclic update algorithms for parameter estimation by pattern searches. *Neural Processing Letters*, 17(2):191–203, 2003. doi: 10.1023/A:1023655202546.

A. Honkela, H. Valpola, A. Ilin, and J. Karhunen. Blind separation of nonlinear mixtures by variational Bayesian learning. *Digital Signal Processing*, 17(5):914–934, 2007. doi: 10.1016/j.dsp.2007.02.009.

A. Honkela, M. Tornio, T. Raiko, and J. Karhunen. Natural conjugate gradient in variational inference. In *Proceedings of the 14th International Conference on Neural Information Processing (ICONIP 2007)*, volume 4985 of *Lecture Notes in Computer Science*, pages 305–314, Kitakyushu, Japan, 2008. Springer-Verlag, Berlin. doi: 10.1007/978-3-540-69162-4_32.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In M. Jordan, editor, *Learning in Graphical Models*, pages 105–161. The MIT Press, Cambridge, MA, USA, 1999.

M. Kuusela, T. Raiko, A. Honkela, and J. Karhunen. A gradient-based algorithm competitive with variational Bayesian EM for mixture of Gaussians. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2009*, pages 1688–1695, Atlanta, GA, USA, June 2009. doi: 10.1109/IJCNN.2009.5178726.

H. Lappalainen and A. Honkela. Bayesian nonlinear independent component analysis by multilayer perceptrons. In M. Girolami, editor, *Advances in Independent Component Analysis*, pages 93–121. Springer-Verlag, Berlin, 2000.

H. Lappalainen and J. Miskin. Ensemble learning. In M. Girolami, editor, *Advances in Independent Component Analysis*, pages 75–92. Springer-Verlag, Berlin, 2000.

M. K. Murray and J. W. Rice. *Differential Geometry and Statistics*. Chapman & Hall, London, 1993.

J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, 1:199–242, 1992. doi: 10.1017/S0962492900002270.

M. Opper and C. Archambeau. The variational Gaussian approximation revisited. *Neural Computation*, 21(3):786–792, 2009. doi: 10.1162/neco.2008.08-07-592.

M. J. D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12(1):241–254, 1977. doi: 10.1007/BF01593790.

T. Raiko, H. Valpola, M. Harva, and J. Karhunen. Building blocks for variational Bayesian learning of latent variable models. *Journal of Machine Learning Research*, 8(Jan):155–201, January 2007.

R. Salakhutdinov and S. T. Roweis. Adaptive overrelaxed bound optimization methods. In *Proc. 20th International Conference on Machine Learning (ICML 2003)*, pages 664–671. AAAI Press, 2003.

M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001. doi: 10.1162/089976601750265045.

M. Seeger. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 603–609. MIT Press, Cambridge, MA, USA, 2000.

S. T. Smith. *Geometric Optimization Methods for Adaptive Filtering*. PhD thesis, Harvard University, Cambridge, MA, USA, 1993.

T. Tanaka. Information geometry of mean-field approximation. In Manfred Opper and David Saad, editors, *Advanced Mean Field Methods: Theory and Practice*, pages 259–273. The MIT Press, Cambridge, MA, USA, 2001.

M. K. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Workshop on Artificial Intelligence and Statistics*, volume 9, pages 844–851, Chia Laguna Resort, Sardinia, Italy, 2010. JMLR W&CP 9.

H. Valpola. *Bayesian Ensemble Learning for Nonlinear Factor Analysis*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 2000. Published in Acta Polytechnica Scandinavica, Mathematics and Computing Series No. 108.

H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002. doi: 10.1162/089976602760408017.

H. Valpola, M. Harva, and J. Karhunen. Hierarchical models of variance sources. *Signal Processing*, 84(2):267–282, 2004. doi: 10.1016/j.sigpro.2003.10.014.

J. Winn and C. M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6: 661–694, April 2005.

Journal of Machine Learning Research 11 (2010) 3269-3311 Submitted 11/08; Revised 6/10; Published 12/10

Classification with Incomplete Data Using Dirichlet Process Priors

Chunping Wang CW36@EE.DUKE.EDU
Xuejun Liao XJLIAO@EE.DUKE.EDU
Lawrence Carin LCARIN@EE.DUKE.EDU
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708-0291, USA

David B. Dunson DUNSON@STAT.DUKE.EDU
Department of Statistical Science
Duke University
Durham, NC 27708-0291, USA

Editor: David Blei

Abstract

A non-parametric hierarchical Bayesian framework is developed for designing a classifier, based on a mixture of simple (linear) classifiers. Each simple classifier is termed a local "expert", and the number of experts and their construction are manifested via a Dirichlet process formulation. The simple form of the "experts" allows analytical handling of incomplete data. The model is extended to allow simultaneous design of classifiers on multiple data sets, termed multi-task learning, with this also performed non-parametrically via the Dirichlet process. Fast inference is performed using variational Bayesian (VB) analysis, and example results are presented for several data sets. We also perform inference via Gibbs sampling, to which we compare the VB results.

Keywords: classification, incomplete data, expert, Dirichlet process, variational Bayesian, multi-task learning

1. Introduction

In many applications one must deal with data that have been collected incompletely. For example, in censuses and surveys, some participants may not respond to certain questions (Rubin, 1987); in email spam filtering, server information may be unavailable for emails from external sources (Dick et al., 2008); in medical studies, measurements on some subjects may be partially lost at certain stages of the treatment (Ibrahim, 1990); in DNA analysis, gene-expression microarrays may be incomplete due to insufficient resolution, image corruption, or simply dust or scratches on the slide (Wang et al., 2006); in sensing applications, a subset of sensors may be absent or fail to operate at certain regions (Williams and Carin, 2005). Unlike in semi-supervised learning (Ando and Zhang, 2005) where missing labels (responses) must be addressed, features (inputs) are partially missing in the aforementioned incomplete-data problems. Since most data analysis procedures (for example, regression and classification) are designed for complete data, and cannot be directly applied to incomplete data, the appropriate handling of missing data is challenging.

Traditionally, data are often "completed" by *ad hoc* editing, such as case deletion and single imputation, where feature vectors with missing values are simply discarded or completed with specific values in the initial stage of analysis, before the main inference (for example, mean im-

putation and regression imputation see Schafer and Graham, 2002). Although analysis procedures designed for complete data become applicable after these edits, shortcomings are clear. For case deletion, discarding information is generally inefficient, especially when data are scarce. Secondly, the remaining complete data may be statistically unrepresentative. More importantly, even if the incomplete-data problem is eliminated by ignoring data with missing features in the training phase, it is still inevitable in the test stage since test data cannot be ignored simply because a portion of features are missing. For single imputation, the main concern is that the uncertainty of the missing features is ignored by imputing fixed values.

The work of Rubin (1976) developed a theoretical framework for incomplete-data problems, where widely-cited terminology for missing patterns was first defined. It was proven that ignoring the *missing mechanism* is appropriate (Rubin, 1976) under the *missing at random* (MAR) assumption, meaning that the *missing mechanism* is conditionally independent of the missing features given the observed data. As elaborated later, given the MAR assumption (Dick et al., 2008; Ibrahim, 1990; Williams and Carin, 2005), incomplete data can generally be handled by full maximum likelihood and Bayesian approaches; however, when the *missing mechanism* does depend on the missing values (*missing not at random* or MNAR), a problem-specific model is necessary to describe the *missing mechanism*, and no general approach exists. In this paper, we address missing features under the MAR assumption. Previous work in this setting may be placed into two groups, depending on whether the missing data are handled before algorithm learning or within the algorithm.

For the former, an extra step is required to estimate $p(\boldsymbol{x}^m|\boldsymbol{x}^o)$, conditional distributions of missing values given observed ones, with this step distinct from the main inference algorithm. After $p(\boldsymbol{x}^m|\boldsymbol{x}^o)$ is learned, various imputation methods may be performed. As a Monte Carlo approach, Bayesian multiple imputation (MI) (Rubin, 1987) is widely used, where multiple ($M > 1$) samples from $p(\boldsymbol{x}^m|\boldsymbol{x}^o)$ are imputed to form M "complete" data sets, with the complete-data algorithm applied on each, and results of those imputed data sets combined to yield a final result. The MI method "completes" data sets so that algorithms designed for complete data become applicable. Furthermore, Rubin (1987) showed that MI does not require as many samples as Monte Carlo methods usually do. With a mild Gaussian mixture model (GMM) assumption for the joint distribution of observed and missing data, Williams et al. (2007) managed to analytically integrate out missing values over $p(\boldsymbol{x}^m|\boldsymbol{x}^o)$ and performed essentially infinite imputations. Since explicit imputations are avoided, this method is more efficient than the MI method, as suggested by empirical results (Williams et al., 2007). Other examples of these two-step methods include Williams and Carin (2005), Smola et al. (2005) and Shivaswamy et al. (2006).

The other class of methods explicitly addresses missing values during the model-learning procedure. The work proposed by Chechik et al. (2008) represents a special case, in which no model is assumed for *structurally absent* values; the margin for the support vector machine (SVM) is re-scaled according to the observed features for each instance. Empirical results (Chechik et al., 2008) show that this procedure is comparable to several single-imputation methods when values are *missing at random*. Another recent work (Dick et al., 2008) handles the missing features inside the procedure of learning a support vector machine (SVM), without constraining the distribution of missing features to any specific class. The main concern is that this method can only handle missing features in the training data; however, in many applications one cannot control whether missing values occur in the training or test data.

A widely employed approach for handling missing values within the algorithm involves maximum likelihood (ML) estimation via expectation maximization (EM) (Dempster et al., 1977). Be-

sides the latent variables (e.g., mixture component indicators), the missing features are also integrated out in the E-step so that the likelihood is maximized with respect to model parameters in the M-step. The main difficulty is that the integral in the E-step is analytically tractable only when an assumption is made on the distribution of the missing features. For example, the intractable integral is avoided by requiring the features to be discrete (Ibrahim, 1990), or assuming a Gaussian mixture model (GMM) for the features (Ghahramani and Jordan, 1994; Liao et al., 2007). The discreteness requirement is often too restrictive, while the GMM assumption is mild since it is well known that a GMM can approximate arbitrary continuous distributions.

In Liao et al. (2007) the authors proposed a quadratically gated mixture of experts (QGME) where the GMM is used to form the gating network, statistically partitioning the feature space into quadratic subregions. In each subregion, one linear classifier works as a local "expert". As a mixture of experts (Jacobs et al., 1991), the QGME is capable of addressing a classification problem with a nonlinear decision boundary in terms of multiple local experts; the simple form of this model makes it straightforward to handle incomplete data without completing kernel functions (Graepel, 2002; Williams and Carin, 2005). However, as in many mixture-of-expert models (Jacobs et al., 1991; Waterhouse and Robinson, 1994; Xu et al., 1995), the number of local experts in the QGME must be specified initially, and thus a model-selection stage is in general necessary. Moreover, since the expectation-maximization method renders a point (single) solution that maximizes the likelihood, over-fitting may occur when data are scarce relative to the model complexity.

In this paper, we first extend the finite QGME (Liao et al., 2007) to an infinite QGME (iQGME), with theoretically an infinite number of experts realized via a Dirichlet process (DP) (Ferguson, 1973) prior; this yields a fully Bayesian solution, rather than a point estimate. In this manner model selection is avoided and the uncertainty on the number of experts is captured in the posterior density function.

The Dirichlet process (Ferguson, 1973) has been an active topic in many applications since the middle 1990s, for example, density estimation (Escobar and West, 1995; MacEachern and Müller, 1998; Dunson et al., 2007) and regression/curve fitting (Müller et al., 1996; Rasmussen and Ghahramani, 2002; Meeds and Osindero, 2006; Shahbaba and Neal, 2009; Rodríguez et al., 2009; Hannah et al., 2010). The latter group is relevant to classification problems of interest in this paper. The work in Müller et al. (1996) jointly modeled inputs and responses as a Dirichlet process mixture of multivariate normals, while Rodríguez et al. (2009) extended this model to simultaneously estimate multiple curves using dependent DP. In Rasmussen and Ghahramani (2002) and Meeds and Osindero (2006) two approaches to constructing infinite mixtures of Gaussian Process (GP) experts were proposed. The difference is that Meeds and Osindero (2006) specified the gating network using a multivariate Gaussian mixture instead of a (fixed) input-dependent Dirichlet Process. In Shahbaba and Neal (2009) another form of infinite mixtures of experts was proposed, where experts are specified by a multinomial logit (MNL) model (also called softmax) and the gating network is Gaussian mixture model with independent covariates. Further, Hannah et al. (2010) generalized existing DP-based nonparametric regression models to accommodate different types of covariates and responses, and further gave theoretical guarantees for this class of models.

Our focus in this paper is on developing classification models that handle incomplete inputs/covariates efficiently using Dirichlet process. Some of the above Dirichlet process regression models are potentially capable of handling incomplete inputs/features; however, none of them actually deal with such problems. In Müller et al. (1996), although the joint multivariate normal assumption over inputs and responses endow this approach with the potential of handling missing

features and/or missing responses naturally, a good estimation for the joint distribution does not guarantee a good estimation for classification boundaries. Other than a full joint Gaussian distribution assumption, explicit classifiers were used to model the conditional distribution of responses given covariates in the models proposed in Meeds and Osindero (2006) and Shahbaba and Neal (2009). These two models are highly related to the iQGME proposed here. The independence assumption of covariates in Shahbaba and Neal (2009) leads to efficient computation but is not appealing for handling missing features. With Gaussian process experts (Meeds and Osindero, 2006), the inference for missing features is not analytical for fast inference algorithms such as variational Bayesian (Beal, 2003) and EM, and the computation could be prohibitive for large data sets. The iQGME seeks a balance between the ease of inference, computational burden and the ability of handling missing features. For high-dimensional data sets, we develop a variant of our model based on mixtures of factor analyzers (MFA) (Ghahramani and Hinton, 1996; Ghahramani and Beal, 2000), where a low-rank assumption is made for the covariance matrices of high-dimensional inputs in each cluster.

In addition to challenges with incomplete data, one must often address an insufficient quantity of labeled data. In Williams et al. (2007) the authors employed semi-supervised learning (Zhu, 2005) to address this challenge, using the contextual information in the unlabeled data to augment the limited labeled data, all done in the presence of missing/incomplete data. Another form of context one may employ to address limited labeled data is multi-task learning (MTL) (Caruana, 1997; Ando and Zhang, 2005), which allows the learning of multiple tasks simultaneously to improve generalization performance. The work of Caruana (1997) provided an overview of MTL and demonstrated it on multiple problems. In recent research, a hierarchical statistical structure has been favored for such models, where information is transferred via a common prior within a hierarchical Bayesian model (Yu et al., 2003; Zhang et al., 2006). Specifically, information may be transferred among related tasks (Xue et al., 2007) when the Dirichlet process (DP) (Ferguson, 1973) is introduced as a common prior. To the best of our knowledge, there is no previous example of addressing incomplete data in a multi-task setting, this problem constituting an important aspect of this paper.

The main contributions of this paper may be summarized as follows. The problem of missing data in classifier design is addressed by extending QGME (Liao et al., 2007) to a fully Bayesian setting, with the number of local experts inferred automatically via a DP prior. The algorithm is further extended to a multi-task setting, again using a non-parametric Bayesian model, simultaneously learning J missing-data classification problems, with appropriate sharing (could be global or local). Throughout, efficient inference is implemented via the variational Bayesian (VB) method (Beal, 2003). To quantify the accuracy of the VB results, we also perform comparative studies based on Gibbs sampling.

The remainder of the paper is organized as follows. In Section 2 we extend the finite QGME (Liao et al., 2007) to an infinite QGME via a Dirichlet process prior. The incomplete-data problem is defined and discussed in Section 3. Extension to the multi-task learning case is considered in Section 4, and variational Bayesian inference is developed in Section 5. Experimental results for synthetic data and multiple real data sets are presented in Section 6, followed in Section 7 by conclusions and a discussions of future research directions.

2. Infinite Quadratically Gated Mixture of Experts

In this section, we first provide a brief review of the quadratically gated mixture of experts (QGME) (Liao et al., 2007) and Dirichlet process (DP) (Ferguson, 1973), and then extend the number of experts to be infinite via DP.

2.1 Quadratically Gated Mixture of Experts

Consider a binary classification problem with real-valued P-dimensional column feature vectors $\boldsymbol{x}_i$ and corresponding class labels $y_i \in \{1,-1\}$. We assume binary labels for simplicity, while the proposed method may be directly extended to cases with more than two classes. Latent variables t_i are introduced as "soft labels" associated with y_i, as in probit models (Albert and Chib, 1993), where $y_i = 1$ if $t_i > 0$ and $y_i = -1$ if $t_i \leq 0$. The finite quadratically gated mixture of experts (QGME) (Liao et al., 2007) is defined as

$$(t_i|z_i = h) \sim \mathcal{N}(\boldsymbol{w}_h^T \boldsymbol{x}_i^b, 1), \tag{1}$$

$$(\boldsymbol{x}_i|z_i = h) \sim \mathcal{N}_P(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h^{-1}), \tag{2}$$

$$(z_i|\boldsymbol{\pi}) \sim \sum_{h=1}^{K} \pi_h \delta_h, \tag{3}$$

with $\sum_{h=1}^{K} \pi_h = 1$, and where δ_h is a point measure concentrated at h (with probability one, a draw from δ_h will be h). The $(P+1) \times K$ matrix $\boldsymbol{W}$ has columns $\boldsymbol{w}_h$, where each $\boldsymbol{w}_h$ are the weights on a local linear classifier, and the $\boldsymbol{x}_i^b$ are feature vectors with an intercept, that is, $\boldsymbol{x}_i^b = [\boldsymbol{x}_i^T, 1]^T$. A total of K groups of $\boldsymbol{w}_h$ are introduced to parameterize the K experts. With probability π_h the indicator for the ith data point satisfies $z_i = h$, which means the hth local expert is selected, and $\boldsymbol{x}_i$ is distributed according to a P-variate Gaussian distribution with mean $\boldsymbol{\mu}_h$ and precision $\boldsymbol{\Lambda}_h$.

It can be seen that the QGME is highly related to the mixture of experts (ME) (Jacobs et al., 1991) and the hierarchical mixture of experts (HME) (Jordan and Jacobs, 1994) if we write the conditional distribution of labels as

$$p(y_i|\boldsymbol{x}_i) = \sum_{h=1}^{K} p(z_i = h|\boldsymbol{x}_i) p(y_i|z_i = h, \boldsymbol{x}_i), \tag{4}$$

where

$$p(y_i|z_i = h, \boldsymbol{x}_i) = \int_{t_i y_i > 0} \mathcal{N}(t_i|\boldsymbol{w}_h^T \boldsymbol{x}_i^b, 1) dt_i, \tag{5}$$

$$p(z_i = h|\boldsymbol{x}_i) = \frac{\pi_h \mathcal{N}_P(\boldsymbol{x}_i|\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h^{-1})}{\sum_{k=1}^{K} \pi_k \mathcal{N}_P(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})}. \tag{6}$$

From (4), as a special case of the ME, the QGME is capable of handling nonlinear problems with linear experts characterized in (5). However, unlike other ME models, the QGME probabilistically partitions the feature space through a mixture of K Gaussian distributions for $\boldsymbol{x}_i$ as in (6). This assumption on the distribution of $\boldsymbol{x}_i$ is mild since it is well known that a Gaussian mixture model (GMM) is general enough to approximate any continuous distribution. In the QGME, $\boldsymbol{x}_i$ as well as y_i are treated as random variables (generative model) and we consider a joint probability $p(y_i, \boldsymbol{x}_i)$ instead of a conditional probability $p(y_i|\boldsymbol{x}_i)$ for fixed $\boldsymbol{x}_i$ as in most ME models (which are typically

discriminative). Previous work on the comparison between discriminative and generative models may be found in Ng and Jordan (2002) and Liang and Jordan (2008). In the QGME, the GMM of the inputs $\boldsymbol{x}_i$ plays two important roles: (*i*) as a gating network, while (*ii*) enabling analytic incorporation of incomplete data during classifier inference (as discussed further below).

The QGME (Liao et al., 2007) is inferred via the expectation-maximization (EM) method, which renders a point-estimate solution for an initially specified model (1)-(3), with a fixed number K of local experts. Since learning the correct model requires model selection, and moreover in many applications there may exist no such fixed "correct" model, in the work reported here we infer the full posterior for a QGME model with the number of experts data-driven. The objective can be achieved by imposing a nonparametric Dirichlet process (DP) prior.

2.2 Dirichlet Process

The Dirichlet process (DP) (Ferguson, 1973) is a random measure defined on measures of random variables, denoted as $\mathcal{DP}(\alpha G_0)$, with a real scaling parameter $\alpha \geq 0$ and a base measure G_0. Assuming that a measure is drawn $G \sim \mathcal{DP}(\alpha G_0)$, the base measure G_0 reflects the prior expectation of G and the scaling parameter α controls how much G is allowed to deviate from G_0. In the limit $\alpha \to \infty$, G goes to G_0; in the limit $\alpha \to 0$, G reduces to a delta function at a random point in the support of G_0.

The stick-breaking construction (Sethuraman, 1994) provides an explicit form of a draw from a DP prior. Specifically, it has been proven that a draw G may be constructed as

$$G = \sum_{h=1}^{\infty} \pi_h \delta_{\boldsymbol{\theta}_h^*}, \tag{7}$$

with $0 \leq \pi_h \leq 1$ and $\sum_{h=1}^{\infty} \pi_h = 1$, and

$$\pi_h = V_h \prod_{l=1}^{h-1} (1 - V_l), \quad V_h \overset{iid}{\sim} Be(1, \alpha), \quad \boldsymbol{\theta}_h^* \overset{iid}{\sim} G_0.$$

From (7), it is clear that G is discrete (with probability one) with an infinite set of weights π_h at atoms $\boldsymbol{\theta}_h^*$. Since the weights π_h decrease stochastically with h, the summation in (7) may be truncated with N terms, yielding an N-level truncated approximation to a draw from the Dirichlet process (Ishwaran and James, 2001).

Assuming that underlying variables $\boldsymbol{\theta}_i$ are drawn i.i.d. from G, the associated data $\chi_i \sim F(\boldsymbol{\theta}_i)$ will naturally cluster with $\boldsymbol{\theta}_i$ taking distinct values $\boldsymbol{\theta}_h^*$, where the function $F(\boldsymbol{\theta})$ represents an arbitrary parametric model for the observed data, with hidden parameters $\boldsymbol{\theta}$. Therefore, the number of clusters is automatically determined by the data and could be "infinite" in principle. Since $\boldsymbol{\theta}_i$ take distinct values $\boldsymbol{\theta}_h^*$ with probabilities π_h, this clustering is a statistical procedure instead of a hard partition, and thus we only have a belief on the number of clusters, which is affected by the scaling parameter α. As the value of α influences the prior belief on the clustering, a gamma hyper-prior is usually employed on α.

2.3 Infinite QGME via DP

Consider a classification task with a training data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$, where $\boldsymbol{x}_i \in \mathbb{R}^P$ and $y_i \in \{-1, 1\}$. With soft labels t_i introduced as in Section 2.1, the infinite QGME (iQGME)

model is achieved via a DP prior imposed on the measure G of $(\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i, \boldsymbol{w}_i)$, the hidden variables characterizing the density function of each data point $(\boldsymbol{x}_i, t_i)$. For simplicity, the same symbols are used to denote parameters associated with each data point and the distinct values, with subscripts i and h indexing data points and unique values, respectively:

$$\begin{aligned}(\boldsymbol{x}_i, t_i) &\sim \mathcal{N}_P(\boldsymbol{x}_i|\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i^{-1})\mathcal{N}(t_i|\boldsymbol{w}_i^T\boldsymbol{x}_i^b, 1),\\ (\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i, \boldsymbol{w}_i) &\overset{iid}{\sim} G,\\ G &\sim \mathcal{DP}(\alpha G_0),\end{aligned} \tag{8}$$

where the base measure G_0 is factorized as the product of a normal-Wishart prior for $(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h)$ and a normal prior for $\boldsymbol{w}_h$, for the sake of conjugacy. As discussed in Section 2.2, data samples cluster automatically, and the same mean $\boldsymbol{\mu}_h$, covariance matrix $\boldsymbol{\Lambda}_h$ and regression coefficients (expert) $\boldsymbol{w}_h$ are shared for a given cluster h. Using the stick-breaking construction, we elaborate (8) as follows for $i = 1, \ldots, n$ and $h = 1, \ldots, \infty$:

Data generation:

$$\begin{aligned}(t_i|z_i = h) &\sim \mathcal{N}(\boldsymbol{w}_h^T\boldsymbol{x}_i^b, 1),\\ (\boldsymbol{x}_i|z_i = h) &\sim \mathcal{N}_P(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h^{-1}),\end{aligned}$$

Drawing indicators:

$$\begin{aligned}z_i &\sim \sum_{h=1}^{\infty}\pi_h\delta_h, \quad \text{where} \quad \pi_h = V_h\prod_{l<h}(1-V_l),\\ V_h &\sim Be(1, \alpha),\end{aligned}$$

Drawing parameters from G_0 :

$$\begin{aligned}(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h) &\sim \mathcal{N}_P(\boldsymbol{\mu}_h|\boldsymbol{m}_0, u_0^{-1}\boldsymbol{\Lambda}_h^{-1})\mathcal{W}(\boldsymbol{\Lambda}_h|\boldsymbol{B}_0, \nu_0),\\ \boldsymbol{w}_h &\sim \mathcal{N}_{P+1}(\boldsymbol{\zeta}, [\mathrm{diag}(\boldsymbol{\lambda})]^{-1}), \quad \text{where} \quad \boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_{P+1}].\end{aligned}$$

Furthermore, to achieve a more robust algorithm, we assign diffuse hyper-priors on several crucial parameters. As discussed in Section 2.2, the scaling parameter α reflects our prior belief on the number of clusters. For the sake of conjugacy, a diffuse Gamma prior is usually assumed for α as suggested by West et al. (1994). In addition, parameters $\boldsymbol{\zeta}, \boldsymbol{\lambda}$ characterizing the prior of the distinct local classifiers $\boldsymbol{w}_h$ are another set of important parameters, since we focus on classification tasks. Normal-Gamma priors are the conjugate priors for the mean and precision of a normal density. Therefore,

$$\begin{aligned}\alpha &\sim Ga(\tau_{10}, \tau_{20}),\\ (\boldsymbol{\zeta}|\boldsymbol{\lambda}) &\sim \mathcal{N}_{P+1}(\boldsymbol{0}, \gamma_0^{-1}[\mathrm{diag}(\boldsymbol{\lambda})]^{-1}),\\ \lambda_p &\sim Ga(a_0, b_0), \quad p = 1, \ldots, P+1,\end{aligned}$$

where $\tau_{10}, \tau_{20}, a_0, b_0$ are usually set to be much less than one and of about the same magnitude, so that the constructed Gamma distributions with means about one and large variances are diffuse; γ_0 is usually set to be around one.

The graphical representation of the iQGME for single-task learning is shown in Figure 1. We notice that a possible variant with sparse local classifiers could be obtained if we impose zero mean for the local classifiers $\boldsymbol{w}_h$, that is, $\boldsymbol{\zeta} = \boldsymbol{0}$, and retain the Gamma hyper-prior for the precision $\boldsymbol{\lambda}$, as

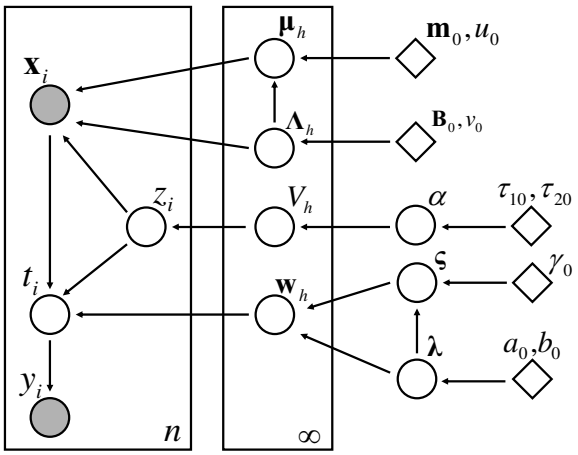


Figure 1: Graphical representation of the iQGME for single-task leaning. All circles denote random variables, with shaded ones indicating observable data, and bright ones representing hidden variables. Diamonds denote fixed hyper-parameters, boxes represent independent replicates with numbers at the lower right corner indicating the numbers of i.i.d. copies, and arrows indicate the dependence between variables (pointing from parents to children).

in the relevance vector machine (RVM) (Tipping, 2000), which employs a corresponding Student-t sparseness prior on the weights. Although this sparseness prior is useful for seeking relevant features in many applications, imposing the same sparse pattern for all the local experts is not desirable.

2.4 Variant for High-Dimensional Problems

For the classification problem, we assume access to a training data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$, where feature vectors $\boldsymbol{x}_i \in \mathbb{R}^P$ and labels $y_i \in \{-1, 1\}$. We have assumed that the feature vectors of objects in cluster h are generated from a P-variate normal distribution with mean $\boldsymbol{\mu}_h$ and covariance matrix $\boldsymbol{\Lambda}_h^{-1}$, that is,

$$(\boldsymbol{x}_i | z_i = h) \quad \sim \quad \mathcal{N}_P(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h^{-1}) \tag{9}$$

It is well known that each covariance matrix has $P(P+1)/2$ parameters to be estimated. Without any further assumption, the estimation of these parameters could be computationally prohibitive for large P, especially when the number of available training data n is small, which is common for classification applications. By imposing an approximately low-rank constraint on the covariances, as in well-studied mixtures of factor analyzers (MFA) models (Ghahramani and Hinton, 1996; Ghahramani and Beal, 2000), the number of unknowns could be significantly reduced. Specifically, assume a vector of standard normal latent factors $\boldsymbol{s}_i \in \mathbb{R}^{T \times 1}$ for data $\boldsymbol{x}_i$, a factor loading matrix $\boldsymbol{A}_h \in \mathbb{R}^{P \times T}$ for cluster h, and Gaussian residues $\boldsymbol{\epsilon}_i$ with diagonal covariance matrix $\psi_h \boldsymbol{I}_P$, then

$$(\boldsymbol{x}_i | z_i = h) \quad \sim \quad \mathcal{N}_P(\boldsymbol{A}_h \boldsymbol{s}_i + \boldsymbol{\mu}_h, \psi_h^{-1} \boldsymbol{I}_P).$$

Marginalizing $\boldsymbol{s}_i$ with $\boldsymbol{s}_i \sim \mathcal{N}_T(\boldsymbol{0}, \boldsymbol{I}_T)$, we recover (9), with $\boldsymbol{\Lambda}_h^{-1} = \boldsymbol{A}_h \boldsymbol{A}_h^T + \psi_h^{-1} \boldsymbol{I}_P$. The number of free parameters is significantly reduced if $T << P$.

In this paper, we modify the MFA model for classification applications with scarce samples. First, we consider a common loading matrix $\boldsymbol{A}$ for all the clusters, and introduce a binary vector $\boldsymbol{b}_h$ for each cluster to select which columns of $\boldsymbol{A}$ are used, that is,

$$(\boldsymbol{x}_i|z_i=h) \quad \sim \quad \mathcal{N}_P(\boldsymbol{A}\text{diag}(\boldsymbol{d}\circ\boldsymbol{b}_h)\boldsymbol{s}_i+\boldsymbol{\mu}_h,\psi_h^{-1}\boldsymbol{I}_P),$$

where each column of $\boldsymbol{A}$, $\boldsymbol{A}_l \sim \mathcal{N}_P(\boldsymbol{0},P^{-1}\boldsymbol{I}_P)$, $\boldsymbol{s}_i \sim \mathcal{N}_L(\boldsymbol{0},\boldsymbol{I}_L)$, $\boldsymbol{d}$ is a vector responsible for scale, and $\circ$ is a component-wise (Hadamard) product. For $\boldsymbol{d}$ we employ the prior $d_l \sim \mathcal{N}(0,\beta_l^{-1})$ with $\beta_l \sim Ga(c_0,d_0)$. Furthermore, we let the algorithm infer the intrinsic number of factors by imposing a low-rank belief for each cluster through the prior of $\boldsymbol{b}_h$, that is,

$$b_{hl} \quad \sim \quad Bern(\pi_{hl}), \quad \pi_{hl} \sim Be(a_0/L, b_0(L-1)/L), \quad l=1,\ldots,L,$$

where L is a large number, which defines the largest possible dimensionality the algorithm may infer. Through the choice of a_0 and b_0 we impose our prior belief about the intrinsic dimensionality of cluster h (upon integrating out the draw $\boldsymbol{\pi}_h$, the number of non-zero components of $\boldsymbol{b}_h$ is drawn from $Binomial[L, a_0/(a_0+b_0(L-1))]$). As a result, both the number of clusters and the dimensionality of each cluster is inferred by this variant of iQGME.

With this form of iQGME, we could build local linear classifiers in either the original feature space or the (low-dimensional) space of latent factors $\boldsymbol{s}_i$. For the sake of computational simplicity, we choose to classify in the low-dimensional factor space.

3. Incomplete Data Problem

In the above discussion it was assumed that all components of the feature vectors were available (no missing data). In this section, we consider the situation for which feature vectors $\boldsymbol{x}_i$ are partially observed. We partition each feature vector $\boldsymbol{x}_i$ into observed and missing parts, $\boldsymbol{x}_i = [\boldsymbol{x}_i^{o_i};\boldsymbol{x}_i^{m_i}]$, where $\boldsymbol{x}_i^{o_i} = \{x_{ip} : p \in o_i\}$ denotes the subvector of observed features and $\boldsymbol{x}_i^{m_i} = \{x_{ip} : p \in m_i\}$ represents the subvector of missing features, with o_i and m_i denoting the set of indices for observed and missing features, respectively. Each $\boldsymbol{x}_i$ has its own observed set o_i and missing set m_i, which may be different for each i. Following a generic notation (Schafer and Graham, 2002), we refer to $\boldsymbol{R}$ as the missingness. For an arbitrary missing pattern, $\boldsymbol{R}$ could be defined as a missing data indicator matrix, that is,

$$R_{ip} = \begin{cases} 1, & x_{ip} \text{ observed,} \\ 0, & x_{ip} \text{ missing.} \end{cases}$$

We use $\boldsymbol{\xi}$ to denote parameters characterizing the distribution of $\boldsymbol{R}$, which is usually called the *missing mechanism*. In the classification context, the joint distribution of class labels, observed features and the missingness $\boldsymbol{R}$ may be given by integrating out the missing features $\boldsymbol{x}^m$,

$$p(y,\boldsymbol{x}^o,\boldsymbol{R}|\boldsymbol{\theta},\boldsymbol{\xi}) = \int p(y,\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{R}|\boldsymbol{x},\boldsymbol{\xi})d\boldsymbol{x}^m. \tag{10}$$

To handle such a problem analytically, assumptions must be made on the distribution of $\boldsymbol{R}$. If the *missing mechanism* is conditionally independent of missing values $\boldsymbol{x}^m$ given the observed data, that is, $p(\boldsymbol{R}|\boldsymbol{x},\boldsymbol{\xi}) = p(\boldsymbol{R}|\boldsymbol{x}^o,\boldsymbol{\xi})$, the missing data are defined to be *missing at random* (MAR) (Rubin, 1976). Consequently, (10) reduces to

$$p(y,\boldsymbol{x}^o,\boldsymbol{R}|\boldsymbol{\theta},\boldsymbol{\xi}) = p(\boldsymbol{R}|\boldsymbol{x}^o,\boldsymbol{\xi})\int p(y,\boldsymbol{x}|\boldsymbol{\theta})d\boldsymbol{x}^m = p(\boldsymbol{R}|\boldsymbol{x}^o,\boldsymbol{\xi})p(y,\boldsymbol{x}^o|\boldsymbol{\theta}). \tag{11}$$

According to (11), the likelihood is factorizable under the assumption of MAR. As long as the prior $p(\boldsymbol{\theta},\boldsymbol{\xi}) = p(\boldsymbol{\theta})p(\boldsymbol{\xi})$ (factorizable), the posterior

$$p(\boldsymbol{\theta},\boldsymbol{\xi}|y,\boldsymbol{x}^o,\boldsymbol{R}) \propto p(y,\boldsymbol{x}^o,\boldsymbol{R}|\boldsymbol{\theta},\boldsymbol{\xi})p(\boldsymbol{\theta},\boldsymbol{\xi}) = p(\boldsymbol{R}|\boldsymbol{x}^o,\boldsymbol{\xi})p(\boldsymbol{\xi})p(y,\boldsymbol{x}^o|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

is also factorizable. For the purpose of inferring model parameters $\boldsymbol{\theta}$, no explicit specification is necessary on the distribution of the missingness. As an important special case of MAR, *missing completely at random* (MCAR) occurs if we can further assume that $p(\boldsymbol{R}|\boldsymbol{x},\boldsymbol{\xi}) = p(\boldsymbol{R}|\boldsymbol{\xi})$, which means the distribution of missingness is independent of observed values $\boldsymbol{x}^o$ as well. When the *missing mechanism* depends on missing values $\boldsymbol{x}^m$, the data are termed to be *missing not at random* (MNAR). From (10), an explicit form has to be assumed for the distribution of the missingness, and both the accuracy and the computational efficiency should be concerned.

When missingness is not totally controlled, as in most realistic applications, we cannot tell from the data alone whether the MCAR or MAR assumption is valid. Since the MCAR or MAR assumption is unlikely to be precisely satisfied in practice, inference based on these assumptions may lead to a bias. However, as demonstrated in many cases, it is believed that for realistic problems departures from MAR are usually not large enough to significantly impact the analysis (Collins et al., 2001). On the other hand, without the MAR assumption, one must explicitly specify a model for the missingness $\boldsymbol{R}$, which is a difficult task in most cases. As a result, the data are typically assumed to be either MCAR or MAR in the literature, unless significant correlations between the missing values and the distribution of the missingness are suspected.

In this work we make the MAR assumption, and thus expression (11) applies. In the iQGME framework, the joint likelihood may be further expanded as

$$p(y,\boldsymbol{x}^o|\boldsymbol{\theta}) = \int p(y,\boldsymbol{x}|\boldsymbol{\theta})d\boldsymbol{x}^m = \int_{ty>0} \int p(t|\boldsymbol{x},\boldsymbol{\theta}_2)p(\boldsymbol{x}|\boldsymbol{\theta}_1)d\boldsymbol{x}^m dt. \tag{12}$$

The solution to such a problem with incomplete data $\boldsymbol{x}^m$ is analytical since the distributions of t and $\boldsymbol{x}$ are assumed to be a Gaussian and a Gaussian mixture model, respectively. Naturally, the missing features could be regarded as hidden variables to be inferred and the graphical representation of the iQGME with incomplete data remains the same as in Figure 1, except that the node presenting features are partially observed now. As elaborated below, the important but mild assumption that the features are distributed as a GMM enables us to analytically infer the variational distributions associated with the missing values in a procedure of variational Bayesian inference.

As in many models (Williams et al., 2007), estimating the distribution of the missing values first and learning the classifier at a second step gives the flexibility of selecting the classifier for the second step. However, (12) suggests that the classifier and the data distribution are coupled, provided that partial data are missing and thus have to be integrated out. Therefore, a joint estimation of missing features and classifiers (searching in the space of $(\boldsymbol{\theta_1},\boldsymbol{\theta}_2)$) is more desirable than a two-step process (searching in the space of $\boldsymbol{\theta_1}$ for the distribution of the data, and then in the space of $\boldsymbol{\theta}_2$ for the classifier).

4. Extension to Multi-Task Learning

Assume we have J data sets, with the jth represented as $\mathcal{D}_j = \{(\boldsymbol{x}_{ji}, y_{ji}) : i = 1,\ldots,n_j\}$; our goal is to design a classifier for each data set, with the design of each classifier termed a "task". One may learn separate classifiers for each of the J data sets (single-task learning) by ignoring connections between

the data sets, or a single classifier may be learned based on the union of all data (pooling) by ignoring differences between the data sets. More appropriately, in a hierarchical Bayesian framework J task-dependent classifiers may be learned jointly, with information borrowed via a higher-level prior (multi-task learning). In some previous research all tasks are assumed to be equally related to each other (Yu et al., 2003; Zhang et al., 2006), or related tasks share exactly the same task-dependent classifier (Xue et al., 2007). With multiple local experts, the proposed iQGME model for a particular task is relatively flexible, enabling the borrowing of information across the J tasks (two data sets may share *parts* of the respective classifiers, without requiring sharing of all classifier components).

As discussed in Section 2.2, a DP prior encourages clustering (each cluster corresponds to a local expert). Now considering multiple tasks, a hierarchical Dirichlet process (HDP) (Teh et al., 2006) may be considered to solve the problem of sharing clusters (local experts) across multiple tasks. Assume a random measure G_j is associated with each task j, where each G_j is an independent draw from Dirichlet process $\mathcal{DP}(\alpha G_0)$ with a base measure G_0 drawn from an upper-level Dirichlet process $\mathcal{DP}(\beta H)$, that is,

$$\begin{aligned} G_j &\sim \mathcal{DP}(\alpha G_0), \quad \text{for} \quad j = 1, \ldots, J, \\ G_0 &\sim \mathcal{DP}(\beta H). \end{aligned}$$

As a draw from a Dirichlet process, G_0 is discrete with probability one and has a stick-breaking representation as in (7). With such a base measure, the task-dependent DPs reuse the atoms $\boldsymbol{\theta}_h^*$ defined in G_0, yielding the desired sharing of atoms among tasks.

With the task-dependent iQGME defined in (8), we consider all J tasks jointly:

$$\begin{aligned} (\boldsymbol{x}_{ji}, t_{ji}) &\sim \mathcal{N}_P(\boldsymbol{x}_{ji}|\boldsymbol{\mu}_{ji}, \boldsymbol{\Lambda}_{ji}^{-1}) \mathcal{N}(t_{ji}|\boldsymbol{w}_{ji}^T \boldsymbol{x}_{ji}^b, 1), \\ (\boldsymbol{\mu}_{ji}, \boldsymbol{\Lambda}_{ji}, \boldsymbol{w}_{ji}) &\overset{iid}{\sim} G_j, \\ G_j &\sim \mathcal{DP}(\alpha G_0), \\ G_0 &\sim \mathcal{DP}(\beta H). \end{aligned}$$

In this form of borrowing information, experts with associated means and precision matrices are shared across tasks as distinct atoms. Since means and precision matrices statistically define local regions in feature space, sharing is encouraged locally. We explicitly write the stick-breaking representations for G_j and G_0, with z_{ji} and c_{jh} introduced as the indicators for each data point and each distinct atom of G_j, respectively. By factorizing the base measure H as a product of a normal-Wishart prior for $(\boldsymbol{\mu}_s, \boldsymbol{\Lambda}_s)$ and a normal prior for $\boldsymbol{w}_s$, the hierarchical model of the multi-

task iQGME via the HDP is represented as

$$
\begin{aligned}
&\text{Data Generation:}\\
(t_{ji}|c_{jh}=s,z_{ji}=h) &\sim \mathcal{N}(\boldsymbol{w}_s^T\boldsymbol{x}_{ji}^b,1),\\
(\boldsymbol{x}_{ji}|c_{jh}=s,z_{ji}=h) &\sim \mathcal{N}_P(\boldsymbol{\mu}_s,\boldsymbol{\Lambda}_s^{-1}),\\
&\text{Drawing lower-level indicators:}\\
z_{ji} &\sim \sum_{h=1}^{\infty}\pi_{jh}\delta_h, \quad \text{where} \quad \pi_{jh}=V_{jh}\prod_{l<h}(1-V_{jl}),\\
V_{jh} &\sim Be(1,\alpha),\\
&\text{Drawing upper-level indicators:}\\
c_{jh} &\sim \sum_{s=1}^{\infty}\eta_s\delta_s, \quad \text{where} \quad \eta_s=U_s\prod_{l<s}(1-U_l),\\
U_s &\sim Be(1,\beta),\\
&\text{Drawing parameters from } H:\\
(\boldsymbol{\mu}_s,\boldsymbol{\Lambda}_s) &\sim \mathcal{N}_P(\boldsymbol{\mu}_s|\boldsymbol{m}_0,u_0^{-1}\boldsymbol{\Lambda}_s^{-1})\mathcal{W}(\boldsymbol{\Lambda}_s|\boldsymbol{B}_0,\nu_0),\\
\boldsymbol{w}_s &\sim \mathcal{N}_{P+1}(\boldsymbol{\zeta},[\mathrm{diag}(\boldsymbol{\lambda})]^{-1}).
\end{aligned}
$$

where $j=1,\ldots,J$ and $i=1,\ldots,n_j$ index tasks and data points in each tasks, respectively; $h=1,\ldots,\infty$ and $s=1,\ldots,\infty$ index atoms for task-dependent G_j and the globally shared base G_0, respectively. Hyper-priors are imposed similarly as in the single-task case:

$$
\begin{aligned}
\alpha &\sim Ga(\tau_{10},\tau_{20}),\\
\beta &\sim Ga(\tau_{30},\tau_{40}),\\
(\boldsymbol{\zeta}|\boldsymbol{\lambda}) &\sim \mathcal{N}_{P+1}(\boldsymbol{0},\gamma_0^{-1}[\mathrm{diag}(\boldsymbol{\lambda})]^{-1}),\\
\lambda_p &\sim Ga(a_0,b_0), \quad p=1,\ldots,P+1,
\end{aligned}
$$

The graphical representation of the iQGME for multi-task learning via the HDP is shown in Figure 2.

5. Variational Bayesian Inference

We initially present the inference formalism for single-task learning, and then discuss the (relatively modest) extensions required for the multi-task case.

5.1 Basic Construction

For simplicity we denote the collection of hidden variables and model parameters as Θ and specified hyper-parameters as Ψ. In a Bayesian framework we are interested in $p(\Theta|\mathcal{D},\Psi)$, the joint posterior distribution of the unknowns given observed data and hyper-parameters. From Bayes' rule,

$$p(\Theta|\mathcal{D},\Psi)=\frac{p(\mathcal{D}|\Theta)p(\Theta|\Psi)}{p(\mathcal{D}|\Psi)},$$

where $p(\mathcal{D}|\Psi)=\int p(\mathcal{D}|\Theta)p(\Theta|\Psi)d\Theta$ is the marginal likelihood that often involves multi-dimensional integrals. Since these integrals are nonanalytical in most cases, the computation of the marginal likelihood is the principal challenge in Bayesian inference. These integrals are circumvented if only a

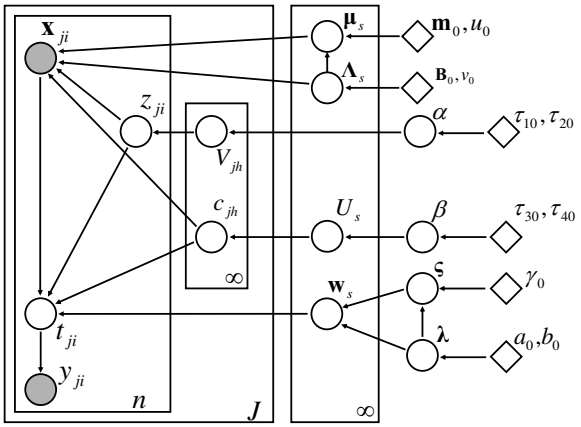


Figure 2: Graphical representation of the iQGME for multi-task leaning via the hierarchical Dirichlet process (HDP). Refer to Figure 1 for additional information.

point estimate $\hat{\Theta}$ is pursued, as in the expectation-maximization algorithm (Dempster et al., 1977). Markov chain Monte Carlo (MCMC) sampling methods (Gelfand et al., 1990; Neal, 1993) provide one class of approximations for the full posterior, based on samples from a Markov chain whose stationary distribution is the posterior of interest. As a Markov chain is guaranteed to converge to its true posterior theoretically as long as the chain is long enough, MCMC samples constitute an unbiased estimation for the posterior. Most previous applications with a Dirichlet process prior (Ishwaran and James, 2001; West et al., 1994), including the related papers we reviewed in Section 1, have been implemented with various MCMC methods. The main concerns of MCMC methods are associated with computational costs for computation of sufficient collection samples, and that diagnosis of convergence is often difficult.

As an efficient alternative, the variational Bayesian (VB) method (Beal, 2003) approximates the true posterior $p(\Theta|\mathcal{D}, \Psi)$ with a variational distribution $q(\Theta)$ with free variational parameters. The problem of computing the posterior is reformulated as an optimization problem of minimizing the Kullback-Leibler (KL) divergence between $q(\Theta)$ and $p(\Theta|\mathcal{D}, \Psi)$, which is equivalent to maximizing a lower bound of log $p(\mathcal{D}|\Psi)$, the log marginal likelihood. This optimization problem can be solved iteratively with two assumptions on $q(\Theta)$: (*i*) $q(\Theta)$ is factorized; (*ii*) the factorized components of $q(\Theta)$ come from the same exponential family as the corresponding priors do. Since the lower bound cannot achieve the true log marginal likelihood in general, the approximation given by the variational Bayesian method is biased. Another issue concerning the VB algorithm is that the solution may be trapped at local optima since the optimization problem is not convex. The main advantages of VB include the ease of convergence diagnosis and computational efficiency. As the VB is solving an optimization problem, the objective function—the lower bound of the log marginal likelihood—is a natural criterion for convergence diagnosis. Therefore, VB is a good alternative to MCMC when conjugacy is achieved and computational efficiency is desired. In recent publications (Blei and Jordan, 2006; Kurihara et al., 2007), discussions on the implementation of the variational Bayesian inference are given for Dirichlet process mixtures.

We implement the variational Bayesian inference throughout this paper, with comparisons made to Gibbs sampling. Since it is desirable to maintain the dependencies among random variables (e.g.,

shown in the graphical models Figure 1) in the variational distribution $q(\Theta)$, one typically only breaks those dependencies that bring difficulty to computation. In the subsequent inference for the iQGME, we retain some dependencies as unbroken. Following Blei and Jordan (2006), we employ stick-breaking representations with a truncation level N as variational distributions to approximate the infinite-dimensional random measures G.

We detail the variational Bayesian inference for the case of incomplete data. The inference for the complete-data case is similar, except that all feature vectors are fully observed and thus the step of learning missing values is skipped. To avoid repetition, a thorough procedure for the complete-data case is not included, with differences from the incomplete-data case indicated.

5.2 Single-task Learning

For single-task iQGME the unknowns are $\Theta = \{\boldsymbol{t}, \boldsymbol{x}^m, \boldsymbol{z}, \boldsymbol{V}, \alpha, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{W}, \boldsymbol{\zeta}, \boldsymbol{\lambda}\}$, with hyper-parameters $\Psi = \{\boldsymbol{m}_0, u_0, \boldsymbol{B}_0, \nu_0, \tau_{10}, \tau_{20}, \gamma_0, a_0, b_0\}$. We specify the factorized variational distributions as

$$\begin{aligned} & q(\boldsymbol{t}, \boldsymbol{x}^m, \boldsymbol{z}, \boldsymbol{V}, \alpha, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{W}, \boldsymbol{\zeta}, \boldsymbol{\lambda}) \\ = & \prod_{i=1}^{n} [q_{t_i}(t_i) q_{\boldsymbol{x}_i^{m_i}, z_i}(\boldsymbol{x}_i^{m_i}, z_i)] \prod_{h=1}^{N-1} q_{V_h}(V_h) \prod_{h=1}^{N} [q_{\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h}(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h) q_{\boldsymbol{w}_h}(\boldsymbol{w}_h)] \prod_{p=1}^{P+1} q_{\zeta_p, \lambda_p}(\zeta_p, \lambda_p) q_\alpha(\alpha) \end{aligned}$$

where

- $q_{t_i}(t_i)$ is a truncated normal distribution,

  $$t_i \sim \mathcal{TN}(\mu_i^t, 1, y_i t_i > 0), \quad i = 1, \ldots, n,$$

  which means the density function of t_i is assumed to be normal with mean μ_i^t and unit variance for those t_i satisfying $y_i t_i > 0$.

- $q_{\boldsymbol{x}_i^{m_i}, z_i}(\boldsymbol{x}_i^{m_i}, z_i) = q_{\boldsymbol{x}_i^{m_i}}(\boldsymbol{x}_i^{m_i} | z_i) q_{z_i}(z_i)$, where $q_{z_i}(z_i)$ is a multinomial distribution with probabilities $\boldsymbol{\rho}_i$, and there are N possible outcomes, $z_i \sim \mathcal{M}_N(1, \rho_{i1}, \ldots, \rho_{iN}), \quad i = 1, \ldots, n$. Given the associated indicators z_i, since features are assumed to be distributed as a multivariate Gaussian, the distributions of missing values $\boldsymbol{x}_i^{m_i}$ are still Gaussian according to conditional properties of multivariate Gaussian distributions:

  $$(\boldsymbol{x}_i^{m_i} | z_i = h) \sim \mathcal{N}_{|m_i|}(\boldsymbol{m}_h^{m_i|o_i}, \boldsymbol{\Sigma}_h^{m_i|o_i}), \quad i = 1, \ldots, n, \quad h = 1, \ldots, N.$$

  We retain the dependency between $\boldsymbol{x}_i^{m_i}$ and z_i in the variational distribution since the inference is still tractable; for complete data, the variation distribution for $(\boldsymbol{x}_i^{m_i} | z_i = h)$ is not necessary.

- $q_{V_h}(V_h)$ is a beta distribution,

  $$V_h \sim Be(v_{h1}, v_{h2}), \quad h = 1, \ldots, N-1.$$

  Recall that we have a truncation level of N, which implies that the mixture proportions $\pi_h(\boldsymbol{V})$ are equal to zero for $h > N$. Therefore, $q_{V_h}(V_h) = \delta_1$ for $h = N$, and $q_{V_h}(V_h) = \delta_0$ for $h > N$. For $h < N$, V_h has a variational Beta posterior.

- $q_{\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h}(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h)$ is a normal-Wishart distribution,

  $$(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h) \sim \mathcal{N}_P(\boldsymbol{m}_h, u_h^{-1} \boldsymbol{\Lambda}_h^{-1}) \mathcal{W}(\boldsymbol{B}_h, \nu_h), \quad h = 1, \ldots, N.$$

- $q_{\boldsymbol{w}_h}(\boldsymbol{w}_h)$ is a normal distribution,

$$\boldsymbol{w}_h \sim \mathcal{N}_{P+1}(\boldsymbol{\mu}_h^w, \boldsymbol{\Sigma}_h^w), \quad h = 1, \ldots, N.$$

- $q_{\zeta_p,\lambda_p}(\zeta_p, \lambda_p)$ is a normal-gamma distribution,

$$(\zeta_p, \lambda_p) \sim \mathcal{N}(\phi_p, \gamma^{-1}\lambda_p^{-1}) Ga(a_p, b_p), \quad p = 1, \ldots, P+1.$$

- $q_\alpha(\alpha)$ is a Gamma distribution,

$$\alpha \sim Ga(\tau_1, \tau_2).$$

Given the specifications on the variational distributions, a mean-field variational algorithm (Beal, 2003) is developed for the iQGME model. All update equations and derivations for $q(\boldsymbol{x}_i^{m_i}, z_i)$ are included in the Appendix; similar derivations for other random variables are found elsewhere (Xue et al., 2007; Williams et al., 2007). Each variational parameter is re-estimated iteratively conditioned on the current estimate of the others until the lower bound of the log marginal likelihood converges. Although the algorithm yields a bound for any initialization of the variational parameters, different initializations may lead to different bounds. To alleviate this local-maxima problem, one may perform multiple independent runs with random initializations, and choose the run that produces the highest bound on the marginal likelihood. We will elaborate on our initializations in the experiment section.

For simplicity, we omit the subscripts on the variational distributions and henceforth use q to denote any variational distributions. In the following derivations and update equations, we use generic notation $\langle f \rangle_{q(\cdot)}$ to denote $\mathrm{E}_{q(\cdot)}[f]$, the expectation of a function f with respect to variational distributions $q(\cdot)$. The subscript $q(\cdot)$ is dropped when it shares the same arguments with f.

5.3 Multi-task Learning

For multi-task learning much of the inference is highly related to that of single-task learning, as discussed above; in the following we focus only on differences. In the multi-task learning model, the latent variables are $\boldsymbol{\Theta} = \{\boldsymbol{t}, \boldsymbol{x}^m, \boldsymbol{z}, \boldsymbol{V}, \alpha, \boldsymbol{c}, \boldsymbol{U}, \beta, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{W}, \boldsymbol{\zeta}, \boldsymbol{\lambda}\}$, and hyper-parameters are $\boldsymbol{\Psi} = \{\boldsymbol{m}_0, u_0, \boldsymbol{B}_0, \nu_0, \tau_{10}, \tau_{20}, \tau_{30}, \tau_{40}, \gamma_0, a_0, b_0\}$. We specify the factorized variational distributions as

$$\begin{aligned} & q(\boldsymbol{t}, \boldsymbol{x}^m, \boldsymbol{z}, \boldsymbol{V}, \alpha, \boldsymbol{c}, \boldsymbol{U}, \beta, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{W}, \boldsymbol{\zeta}, \boldsymbol{\lambda}) \\ = & \prod_{j=1}^{J} \{ \prod_{i=1}^{n_j} [q(t_{ji}) q(\boldsymbol{x}_{ji}^{m_{ji}}) q(z_{ji})] \prod_{h=1}^{N-1} q(V_{jh}) \prod_{h=1}^{N} q(c_{jh}) \} \prod_{s=1}^{S-1} q(U_s) \\ & \prod_{s=1}^{S} [q(\boldsymbol{\mu}_s, \boldsymbol{\Lambda}_s) q(\boldsymbol{w}_s)] \prod_{p=1}^{P+1} q(\zeta_p, \lambda_p) q(\alpha) q(\beta) \end{aligned}$$

where the variational distributions of $(t_{ji}, V_{jh}, \alpha, \boldsymbol{\mu}_s, \boldsymbol{\Lambda}_s, \boldsymbol{w}_s, \zeta_p, \lambda_p)$ are assumed to be the same as in the single-task learning, while the variational distributions of hidden variables newly introduced for the upper-level Dirichlet process are specified as

- $q(c_{jh})$ for each indicator c_{jh} is a multinomial distribution with probabilities $\boldsymbol{\sigma}_{jh}$,

$$c_{jh} \sim \mathcal{M}_S(1, \sigma_{jh1}, \ldots, \sigma_{jhS}), \quad j = 1, \ldots, J, \quad h = 1, \ldots, N.$$

- $q(U_s)$ for each weight U_s is a Beta distribution,

$$U_s \sim Be(\kappa_{s1}, \kappa_{s2}), \quad s = 1, \ldots, S-1.$$

  Here we have a truncation level of S for the upper-lever DP, which implies that the mixture proportions $\eta_s(\boldsymbol{U})$ are equal to zero for $s > S$. Therefore, $q(U_s) = \delta_1$ for $s = S$, and $q(U_s) = \delta_0$ for $s > S$. For $s < S$, U_s has a variational Beta posterior.

- $q(\beta)$ for the scaling parameter β is a Gamma distribution,

$$\beta \sim Ga(\tau_3, \tau_4).$$

We also note that with a higher-level of hierarchy, the dependency between the missing values $\boldsymbol{x}_{ji}^{m_{ji}}$ and the associated indicator z_{ji} has to be broken so that the inference becomes tractable. The variational distribution of z_{ji} is still assumed to be multinomial distributed, while $\boldsymbol{x}_{ji}^{m_{ji}}$ is assumed to be normally distributed but no longer dependent on z_{ji}. All update equations are included in the Appendix.

5.4 Prediction

For a new observed feature vector $\boldsymbol{x}_\star^{o_\star}$, the prediction on the associated class label $y_\star$ is given by integrating out the missing values.

$$\begin{aligned} P(y_\star = 1|\boldsymbol{x}_\star^{o_\star}, \mathcal{D}) &= \frac{p(y_\star = 1, \boldsymbol{x}_\star^{o_\star}|\mathcal{D})}{p(\boldsymbol{x}_\star^{o_\star}|\mathcal{D})} = \frac{\int p(y_\star = 1, \boldsymbol{x}_\star|\mathcal{D}) d\boldsymbol{x}_\star^{m_\star}}{\int p(\boldsymbol{x}_\star|\mathcal{D}) d\boldsymbol{x}_\star^{m_\star}} \\ = & \frac{\int \sum_{h=1}^N P(z_\star = h|\mathcal{D}) p(x_\star|z_\star = h, \mathcal{D}) P(y_\star = 1|\boldsymbol{x}_\star, z_\star = h, \mathcal{D}) d\boldsymbol{x}_\star^{m_\star}}{\int \sum_{k=1}^N P(z_\star = k|\mathcal{D}) p(x_\star|z_\star = k, \mathcal{D}) d\boldsymbol{x}_\star^{m_\star}}. \end{aligned}$$

We marginalize the hidden variables over their variational distributions to compute the predictive probability of the class label

$$P(y_\star = 1|\boldsymbol{x}_\star^{o_\star}, \mathcal{D}) = \frac{\sum_{h=1}^N \mathrm{E}_V[\pi_h] \int_0^\infty \int \mathrm{E}_{\boldsymbol{\mu}_h \boldsymbol{\Lambda}_h}[\mathcal{N}_P(\boldsymbol{x}_\star|\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h^{-1})] \mathrm{E}_{\boldsymbol{w}_h}\left[\mathcal{N}(t_\star|\boldsymbol{w}_h^T \boldsymbol{x}_\star^b, 1)\right] d\boldsymbol{x}_\star^{m_\star} dt_\star}{\sum_{k=1}^N \mathrm{E}_V[\pi_k] \int \mathrm{E}_{\boldsymbol{\mu}_k \boldsymbol{\Lambda}_k}[\mathcal{N}_P(\boldsymbol{x}_\star|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})] d\boldsymbol{x}_\star^{m_\star}}$$

where

$$\mathrm{E}_V[\pi_h] = \mathrm{E}_V[V_h \prod_{l<h}(1 - V_l)] = \langle V_h \rangle \prod_{l<h} \langle 1 - V_l \rangle = \left[\frac{v_{h1}}{v_{h1} + v_{h2}}\right]^{\mathbf{1}(h<N)} \prod_{l<h} \left[\frac{v_{l2}}{v_{l1} + v_{l2}}\right]^{\mathbf{1}(h>1)}.$$

The expectation $\mathrm{E}_{\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h}[\mathcal{N}_P(\boldsymbol{x}^\star|\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h^{-1})]$ is a multivariate Student-t distribution (Attias, 2000). However, for the incomplete-data situation, the integral over the missing values is tractable only when the two terms in the integral are both normal. To retain the form of norm distributions, we use the posterior means of $\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h$ and $\boldsymbol{w}_h$ to approximate the variables:

$$\begin{aligned} P(y_\star = 1|\boldsymbol{x}_\star^{o_\star}, \mathcal{D}) \quad \approx \quad & \frac{\sum_{h=1}^N \mathrm{E}_V[\pi_h] \int_0^\infty \int \mathcal{N}_P(\boldsymbol{x}_\star|\boldsymbol{m}_h, (\nu_h \boldsymbol{B}_h)^{-1}) \mathcal{N}(t_\star|(\boldsymbol{\mu}_h^w)^T \boldsymbol{x}_\star^b, 1) d\boldsymbol{x}_\star^{m_\star} dt_\star}{\sum_{k=1}^N \mathrm{E}_V[\pi_k] \int \mathcal{N}_P(\boldsymbol{x}_\star|\boldsymbol{m}_k, (\nu_k \boldsymbol{B}_k)^{-1}) d\boldsymbol{x}_\star^{m_\star}} \\ = \quad & \frac{\sum_{h=1}^N \mathrm{E}_V[\pi_h] \mathcal{N}_{|o_\star|}(\boldsymbol{x}_\star^{o_\star}|\boldsymbol{m}_h^{o_\star}, (\nu_h \boldsymbol{B}_h)^{-1, o_\star o_\star}) \int_0^\infty \mathcal{N}(t_\star|\varphi_{\star h}, g_{\star h}) dt_\star}{\sum_{k=1}^N \mathrm{E}_V[\pi_k] \mathcal{N}_{|o_\star|}(\boldsymbol{x}_\star^{o_\star}|\boldsymbol{m}_k^{o_\star}, (\nu_k \boldsymbol{B}_k)^{-1, o_\star o_\star})} \end{aligned}$$

where

$$\begin{aligned}
\varphi_{\star h} &= [\boldsymbol{m}_h^T, 1]\boldsymbol{\mu}_h^w + \boldsymbol{\Gamma}_{\star h}^T (\boldsymbol{\Delta}_h^{o_\star o_\star})^{-1} (\boldsymbol{x}_\star^{o_\star} - \boldsymbol{m}_h^{o_\star}), \\
g_{\star h} &= 1 + (\bar{\boldsymbol{\mu}}_h^w)^T \boldsymbol{\Delta}_h \bar{\boldsymbol{\mu}}_h^w - \boldsymbol{\Gamma}_{\star h}^T (\boldsymbol{\Delta}_h^{o_\star o_\star})^{-1} \boldsymbol{\Gamma}_{\star h}, \\
\boldsymbol{\Gamma}_{\star h} &= \boldsymbol{\Delta}_h^{o_\star o_\star} (\boldsymbol{\mu}_h^w)^{o_\star} + \boldsymbol{\Delta}_h^{o_\star m_\star} (\boldsymbol{\mu}_h^w)^{m_\star}, \\
\bar{\boldsymbol{\mu}}_h^w &= (\boldsymbol{\mu}_h^w)_{1:P}, \\
\boldsymbol{\Delta}_h &= (\nu_h \boldsymbol{B}_h)^{-1}.
\end{aligned}$$

For complete data the integral of missing features is absent, so we take advantage of the full variational posteriors for prediction.

5.5 Computational Complexity

Given the truncation level (or the number of clusters) N, the data dimensionality P, and the number of data points n, we compare the iQGME to closely related DP regression models (Meeds and Osindero, 2006; Shahbaba and Neal, 2009), in terms of the time and memory complexity. The inference of the iQGME with complete data requires inversion of two $P \times P$ matrices (the covariance matrices for the inputs and the local expert) associated with each cluster. Therefore, the time and memory complexity are $O(2NP^3)$ and $O(2NP^2)$, respectively. With incomplete data, since the missing pattern is unique for each data point, the time and memory complexity increase with number of data points, that is, $O(nNP^3)$ and $O(nNP^2)$, respectively. The mixture of Gaussian process experts (Meeds and Osindero, 2006) requires $O(NP^3 + n^3/N)$ computations for each MCMC iteration if the N experts equally divide the data, and the memory complexity is $O(NP^2 + n^2/N)$. In the model proposed by Shahbaba and Neal (2009), no matrix inversion is needed since the covariates are assumed to be independent. The time and memory complexity are $O(NP)$ and $O(NP)$, respectively.

From the aspect of computational complexity, the model in Meeds and Osindero (2006) is restricted by the increase of both dimensionality and data size; while the model proposed in Shahbaba and Neal (2009) is more efficient. Although the proposed model requires more computations for each MCMC iteration than the latter one, we are able to handle missing values naturally, and much more efficiently compared to the former one. Considering the usual number of iterations required by VB (several dozens) and MCMC (thousands or even tens of thousands), our model is even more efficient.

6. Experimental Results

In all the following experiments the hyper-parameters are set as follows: $a_0 = 0.01$, $b_0 = 0.01$, $\gamma_0 = 0.1$, $\tau_{10} = 0.05$, $\tau_{20} = 0.05$, $\tau_{30} = 0.05$, $\tau_{40} = 0.05$, $u_0 = 0.1$, $\nu_0 = P + 2$, and $\boldsymbol{m}_0$ and $\boldsymbol{B}_0$ are set according to sample mean and sample precision, respectively. These parameters have not been optimized for any particular data set (which are all different in form), and the results are relatively insensitive to "reasonable" settings. The truncation levels for the variational distributions are set to be $N = 20$ and $S = 50$. We have found the results insensitive to the truncation level, for values larger than those considered here.

Because of the local-maxima issue associated with VB, initialization of the inferred VB hyper-parameters is often important. We initialize most variational hyper-parameters using the corresponding prior hyper-parameters, which are data-independent. The precision/covariance matrices

$\boldsymbol{B}_h$ and $\boldsymbol{\Sigma}_h^w$ are simply initialized as identity matrices. However, for several other hyper-parameters, we may obtain information for good start points from the data. Specifically, the variational mean of the soft label μ_i^t is initialized by the associated label y_i. A K-means clustering algorithm is implemented on the feature vectors, and the cluster means and identifications for objects are used to initialize the variational mean of the Gaussian means $\boldsymbol{m}_h$ and the indicator probabilities $\boldsymbol{\rho}_i$, respectively. As an alternative, one may randomly initialize $\boldsymbol{m}_h$ and $\boldsymbol{\rho}_i$ multiple times, and select the solution that produces the highest lower bound on the log marginal likelihood. The two approaches work almost equivalently for low-dimensional problems; however, for problems with moderate to high dimensionality, it could be fairly difficult to get a satisfying initialization by making several random trials.

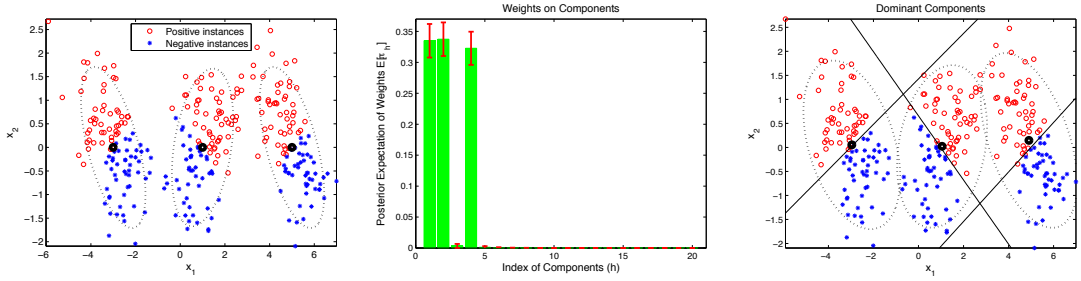
6.1 Synthetic Data

We first demonstrate the proposed iQGME single-task learning model on a synthetic data set, for illustrative purposes. The data are generated according to a GMM model $p(\boldsymbol{x}) = \sum_{k=1}^{3} \pi_k \mathcal{N}_2(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ with the following parameters:

$$\boldsymbol{\pi} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix}, \quad \boldsymbol{\mu}_1 = \begin{bmatrix} -3 \\ 0 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{\mu}_3 = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 0.52 & -0.36 \\ -0.36 & 0.73 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.47 & 0.19 \\ 0.19 & 0.7 \end{bmatrix}, \quad \boldsymbol{\Sigma}_3 = \begin{bmatrix} 0.52 & -0.36 \\ -0.36 & 0.73 \end{bmatrix}.$$

The class boundary for each Gaussian component is given by three lines $x_2 = w_k x_1 + b_k$ for $k = 1,2,3$, where $w_1 = 0.75, b_1 = 2.25$, $w_2 = -0.58, b_2 = 0.58$, and $w_3 = 0.75, b_3 = -3.75$. The simulated data are shown in Figure 3(a), where black dots and dashed ellipses represent the true means and covariance matrices of the Gaussian components, respectively.

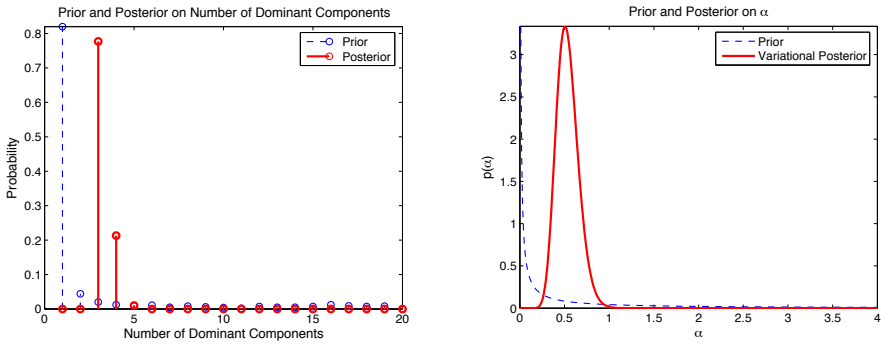


(a) (b) (c)

Figure 3: Synthetic three-Gaussian single-task data with inferred components. (a) Data in feature space with true labels and true Gaussian components indicated; (b) inferred posterior expectation of weights on components, with standard deviations depicted as error bars; (c) ground truth with posterior means of dominant components indicated (the linear classifiers and Gaussian ellipses are inferred from the data).

The inferred mean mixture weights with standard deviations are depicted in Figure 3(b), and it is observed that three dominant mixture components (local "experts") are inferred. The domi-

nant components (those with mean weight larger than 0.005) are characterized by Gaussian means, covariance matrices and local experts, as depicted in Figure 3(c). From Figure 3(c), the nonlinear classification is manifested by using three dominant *local* linear classifiers, with a GMM defining the effective regions stochastically.



(a) (b)

Figure 4: Synthetic three-Gaussian single-task data: (a) prior and posterior beliefs on the number of dominant components; (b) prior and posterior beliefs on α.

An important point is that we are not selecting a "correct" number of mixture components as in most mixture-of-expert models, including the finite QGME model (Liao et al., 2007). Instead, there exists uncertainty on the number of components in our posterior belief. Since this uncertainty is not inferred directly, we obtain samples for the number of dominant components by calculating π_h based on V_h sampled from their probability density functions (prior or variational posterior), and the probability mass functions given by histogram are shown in Figure 4(a). As discussed, the scale parameter α is highly related to the number of clusters, so we depict the prior and the variational posterior on α in Figure 4(b).

The predictions in feature space are presented in Figure 5, where the prediction in sub-figure (a) is given by integrating over the full posteriors of local experts and parameters (means and covariance matrices) of Gaussian components; while the prediction in sub-figure (b) is given by posterior means. We examine these two cases since the analytical integrals over the full posteriors may be unavailable sometimes in practice (for example, for cases with incomplete data as discussed in Section 5). From Figures 5(a) and 5(b), we observe that these two predictions are fairly similar, except that (a) allows more uncertainty on regions with scarce data. The reason for this is that the posteriors are often peaked and thus posterior means are usually representative. As an example, we plot the broad common prior imposed for local experts in Figure 5(c) and the peaked variational posteriors for three dominant experts in Figure 5(d). According to Figure 5, we suggest the usage of full posteriors for prediction whenever integrals are analytical, that is, for experiments with complete data. It also empirically justifies the use of posterior means as an approximation. These results have been computed using VB inference, with MCMC-based results presented below, as a comparison.

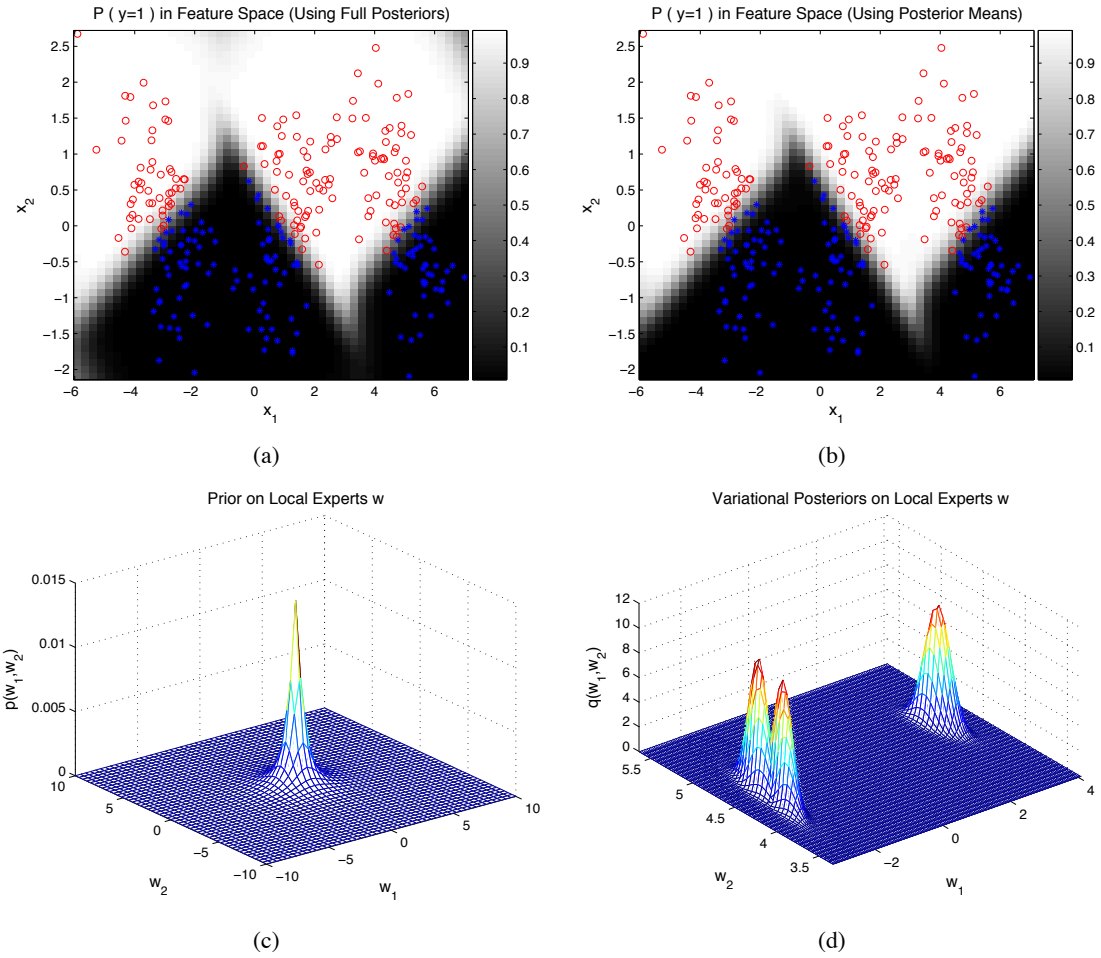


Figure 5: Synthetic three-Gauss single-task data: (a) prediction in feature space using the full posteriors; (b) prediction in feature space using the posterior means; (c) a common broad prior on local experts; (d) variational posteriors on local experts.

6.2 Benchmark Data

To further evaluate the proposed iQGME, we compare it with other models, using benchmark data sets available from the UCI machine learning repository (Newman et al., 1998). Specifically, we consider Wisconsin Diagnostic Breast Cancer (WDBC) and the Johns Hopkins University Ionosphere database (Ionosphere) data sets, which have been studied in the literature (Williams et al., 2007; Liao et al., 2007). These two data sets are summarized in Table 1.

The models we compare to include:

- State-of-the-art classification algorithms: Support Vector Machines (SVM) (Vapnik, 1995) and Relevance Vector Machines (RVM) (Tipping, 2000). We consider both linear models (Linear) and non-linear models with radial basis function (RBF) for both algorithms. For

Data set	Dimension	Number of positive instances	Number of negative instances
Ionosphere	34	126	225
WDBC	30	212	357

Table 1: Details of Ionosphere and WDBC data sets

each data set, the kernel parameter is selected for one training/test/validation separation, and then fixed for all the other experimental settings. The RVM models are implemented with Tipping's Matlab code available at `http://www.miketipping.com/index.php?page=rvm`.

Since those SVM and RVM algorithms are not directly applicable to problems with missing features, we use two methods to impute the missing values before the implementation. One is using the mean of observed values (unconditional mean) for the given feature, referred to as Uncond; the other is using the posterior mean conditional on observed features (conditional mean), referred to as Cond (Schafer and Graham, 2002).

- Classifiers handling missing values: the finite QGME inferred by expectation-maximization (EM) (Liao et al., 2007), referred to as QGME-EM, and a two-stage algorithm (Williams et al., 2007) where the parameters of the GMM for the covariates are estimated first given the observed features, and then a marginalized linear logistic regression (LR) classifier is learned, referred to as LR-Integration. Results are cited from Liao et al. (2007) and Williams et al. (2007), respectively.

In order to simulate the *missing at random* setting, we randomly remove a fraction of feature values according to a uniform distribution, and assume the rest are observed. Any instance with all feature values missing is deleted. After that, we randomly split each data set into training and test subsets, imposing that each subset encompasses at least one instance from each of the classes. Note that the random pattern of missing features and the random partition of training and test subsets are independent of each other. By performing multiple trials we consider the general (average) performance for various data settings. For convenient comparison with Williams et al. (2007) and Liao et al. (2007), the performance of algorithms is evaluated in terms of the area under a receiver operating characteristic (ROC) curve (AUC) (Hanley and McNeil, 1982).

The results on the Ionosphere and WDBC data sets are summarized in Figures 6 and 7, respectively, where we consider 25% and 50% of the feature values missing. Given a portion of missing values, each curve is a function of the fraction of data used in training. For a given size of training data, we perform ten independent trials for the SVM and RVM models and the proposed iQGME.

From both Figures 6 and 7, the proposed iQGME-VB is robust for all the experimental settings, and its overall performance is the best among all algorithms considered. Although the RVM-RBF-Cond and the SVM-RBF-Cond perform well for the Ionosphere data set, especially when the training data is limited, their performance on the WDBC data set is not as good. The kernel methods benefit from the introduction of the RBF kernel for the Ionosphere data set; however, the performance is inferior for the WDBC data set. We also note that the one-step iQGME and the finite QGME outperform the two-step LR-integration. The proposed iQGME consistently performs better than the finite QGME (where, for the latter, in all cases we show results for the best/optimized choice of number of experts K), which reveals the advantage of retaining the uncertainty on the model structure (number of experts) and model parameters. As shown in Figure 7, the advantage of

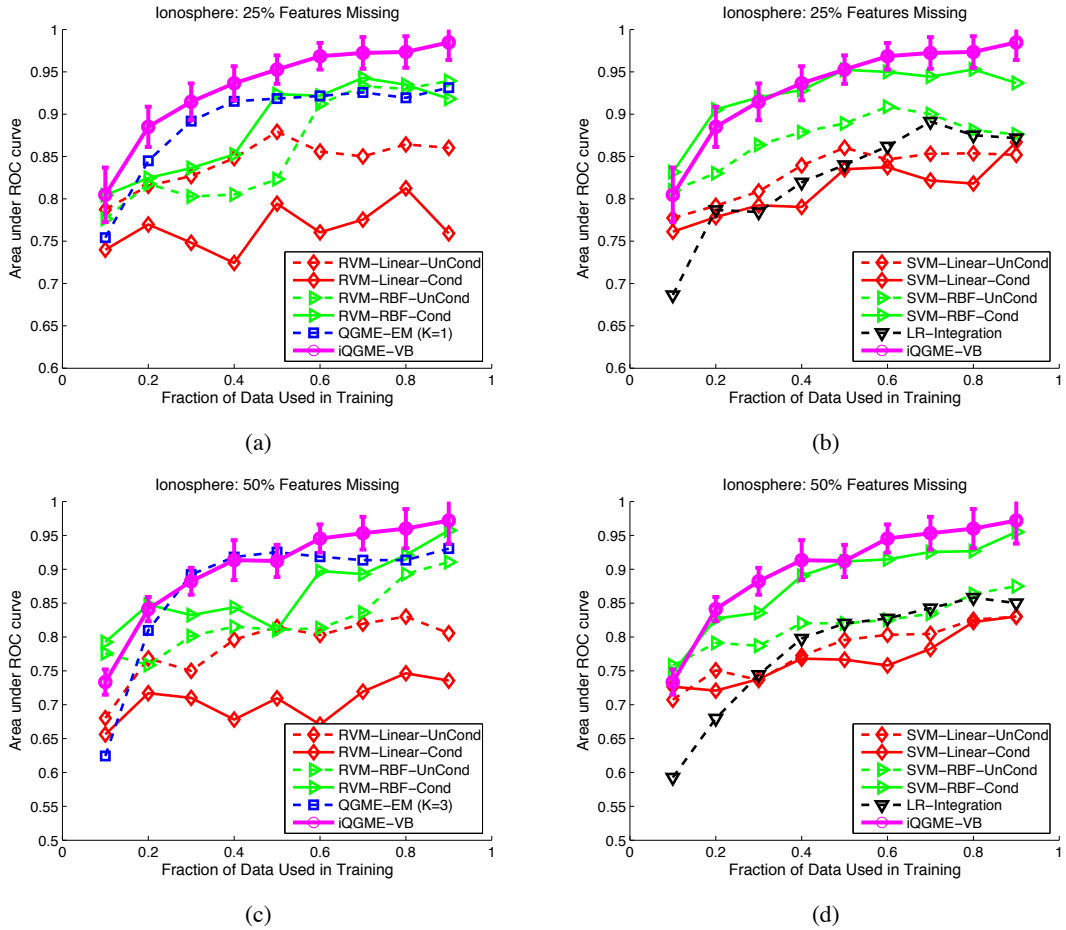
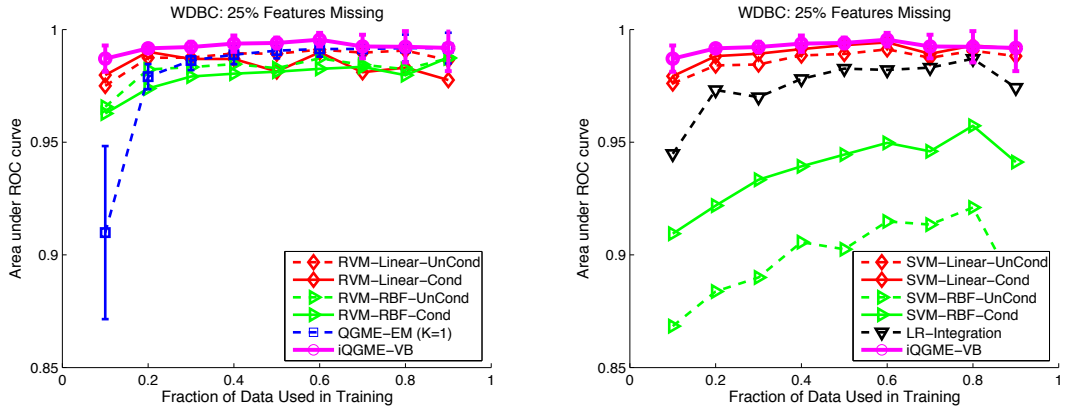


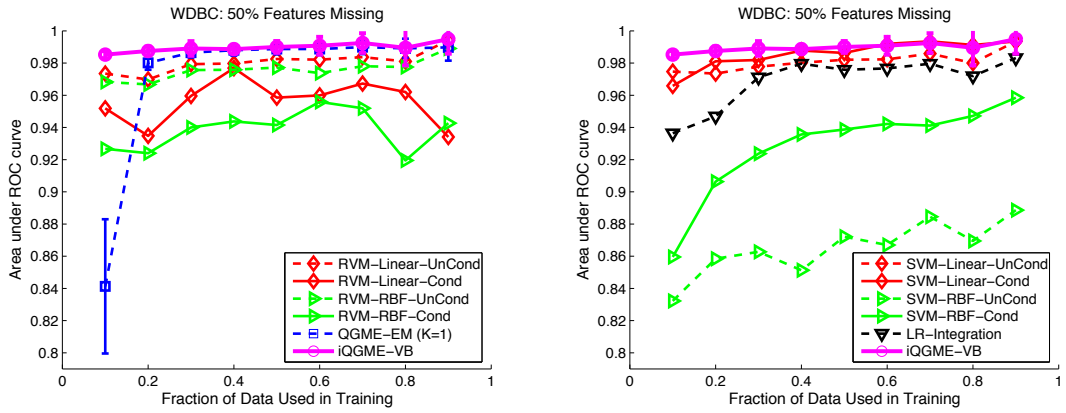
Figure 6: Results on Ionosphere data set for (a)(b) 25%, and (c)(d) 50% of the feature values missing. For legibility, we only report the standard deviation for the proposed iQGME-VB algorithm as error bars, and present the compared algorithms in two figures for each case. The results of the finite QGME solved with an expectation-maximization method are cited from Liao et al. (2007), and those of LR-Integration are cited from Williams et al. (2007). Since the performance of the QGME-EM is affected by the choice of number of experts K, the overall best results among $K = 1, 3, 5, 10, 20$ are cited for comparison in each case (no such selection of K is required for the proposed iQGME-VB algorithm).

considering the uncertainty on the model parameters is fairly pronounced for the WDBC data set, especially when training examples are relatively scarce and thus the point-estimation EM method suffers from over-fitting issues. A more detailed examination on the model uncertainty is shown in Figures 8 and 9.

In Figure 8, the influence of the preset value for K on the QGME-EM model is examined on the Ionosphere data. We observe that with different fractions of missing values and training samples, the values for K which achieve the best performance may be different; as K goes to a large number (e.g., 20 here), the performance gets worse due to over-fitting. In contrast, we do not need to set the



(a)



(c)

Figure 7: Results on WDBC data set for cases when (a)(b) 25%, and (c)(d) 50% of the feature values are missing. Refer to Figure 6 for additional information.

number of clusters for the proposed iQGME-VB model. As long as the truncation level N is large enough ($N = 20$ for all the experiments), the number of clusters is inferred by the algorithm. We give an example for the posterior on the number of clusters inferred by the proposed iQGME-VB model, and report the statistics for the most probable number of experts given each missing fraction and training fraction in Figure 9, which suggests that the number of clusters may vary significantly even for the trials with the same fraction of feature values missing and the same fraction of samples for training. Therefore, it may be not appropriate to set a fixed value for the number of clusters for all the experimental settings as one has to do for the QGME-EM.

Although our main purpose is classification, one may also be interested in how well the algorithm can estimate the missing values while pursuing the main purpose. In Figure 10, we show the ratio of correctly estimated missing values for the Ionosphere data set with 25% feature values missing, where two criteria are considered: true values are one standard deviation (red circles) or two standard deviations (blue squares) away from the posterior means. This figure suggests that the algorithm estimates most of the missing values in a reasonable range away from the true val-

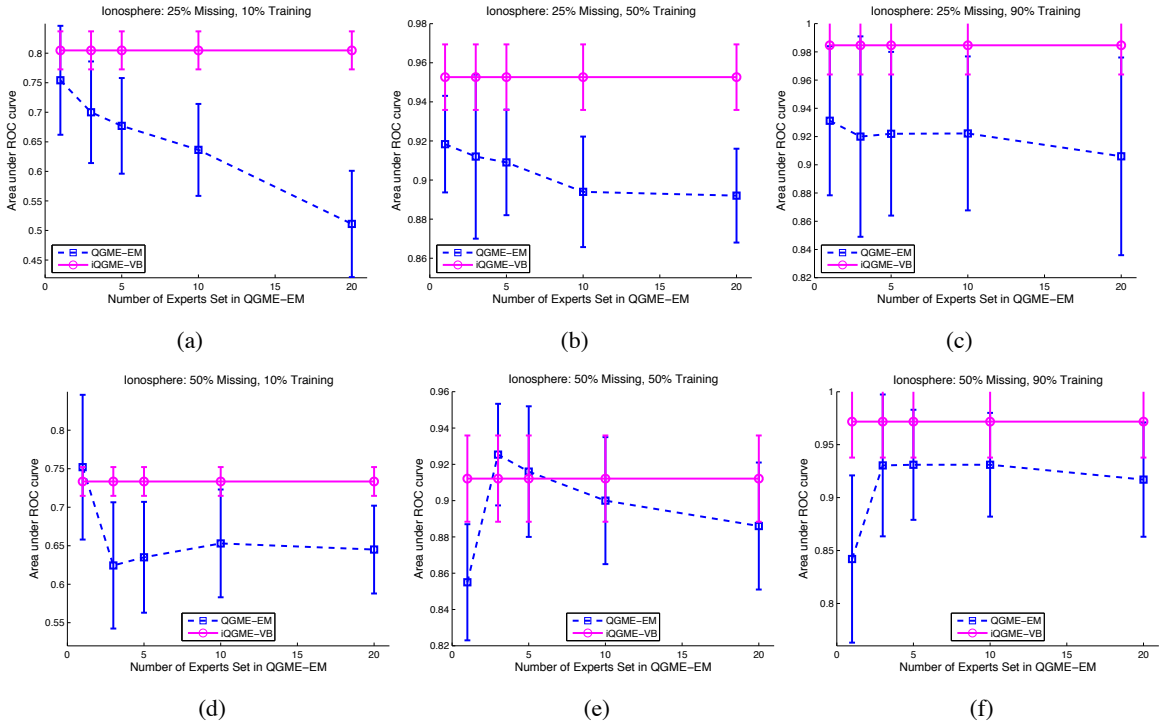
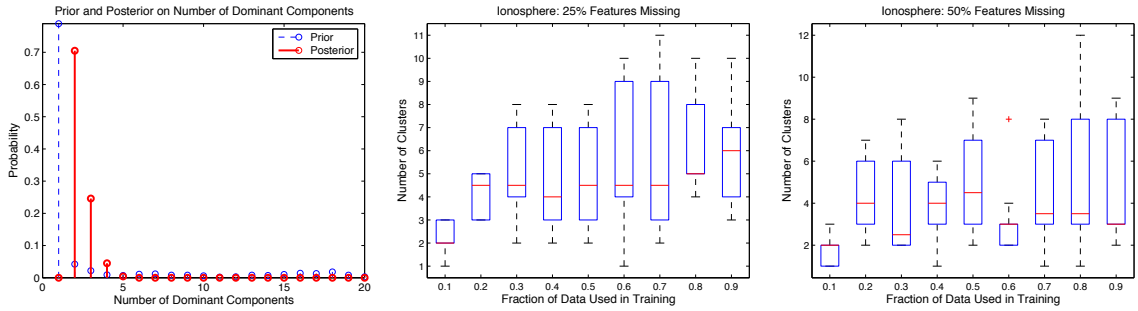


Figure 8: The comparison on the Ionosphere data set between QGME-EM with different preset number of clusters K and the proposed iQGME-VB, when (a)(b)(c) 25%, and (d)(e)(f) 50% of the features are missing. In each row, 10%, 50%, and 90% of samples are used for training, respectively. Results of QGME-EM are cited from Liao et al. (2007).

ues when the training size is large enough; even with not so satisfying estimations (as for limited training data), the classification results are still relatively robust as shown in Figure 6.

We have discussed the advantages and disadvantages for the inference with MCMC and VB in Section 5.1. Here we take the Ionosphere data with 25% features missing as an example to compare these two inference techniques, as shown in Figure 11. It can be seen that they achieve similar performance for the particular iQGME model proposed in this paper. The time consumed for each iteration is also comparable, and increases almost linearly with the training size, as discussed in Section 5.5. The VB inference takes a little bit longer per iteration, probably due to the extra computation for the lower bound of the log marginal likelihood, which serves as convergence criterion. Significant differences occur on the number of iterations we have to take. In the experiment, even though we set a very strict threshold (10^{-6}) for the relative change of the lower bound, the VB algorithm converges at about 50 iterations for most cases except when training data are very scarce (10%). For the MCMC inference, we discard the initial samples from the first 1000 iterations (burn-in), and collect the next 500 samples to present the posterior. It is far from enough to claim convergence; however, we consider it a fair comparison for computation as the two methods yield similar results under this setting. Given the fact that the VB algorithm only takes about $1/30$ the CPU time, and VB and MCMC performance are similar, in the following examples we only present results based on VB inference. However, in all the examples below we also performed Gibbs sam-



(a) (b) (c)

Figure 9: Number of clusters for the Ionosphere data set inferred by iQGME-VB. (a) Prior and inferred posterior on the number of clusters for one trial given 10% samples for training, the number of clusters for the case when (b) 25%, and (c) 50% of features are missing. The most probable value of clusters number is used for each trial to generate (b) and (c) (e.g., the most probable value of clusters number is two for the trial shown in (a)). In (b) and (c), the distribution of number of clusters for the ten trials given each missing fraction and training fraction is presented as a box-plot, where the red line represents the median; the bottom and top of the blue box are the 25th and 75th percentile, respectively; the bottom and top black lines are the end of the whiskers, which could be the minimum and maximum, respectively; if some data are beyond 1.5 times of the length of the blue box (interquartile range), they are outliers, indicated by a red '+'.

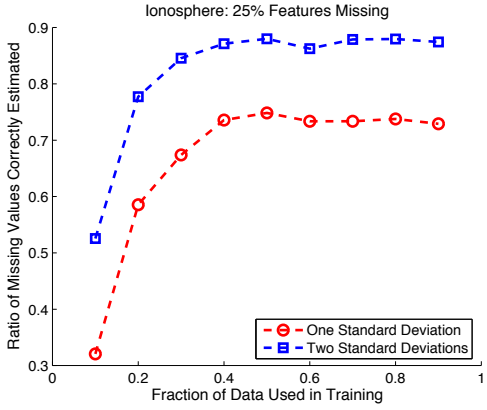
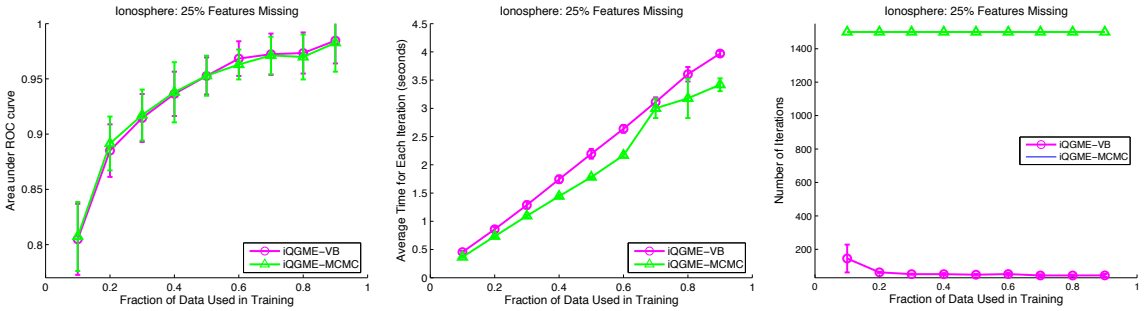


Figure 10: Ratio of missing values whose true values are one standard deviation (red circles) or two standard deviations (blue squares) away from the posterior means for the Ionosphere data set with 25% feature values missing. One trial for each training size is considered.

pling, and the relative inference consistency and computational costs relative to VB were found to be as summarized here (i.e., in all cases there was close agreement between the VB and MCMC inferences, and considerable computational acceleration manifested by VB).



(a) (b) (c)

Figure 11: Comparison between VB and MCMC inferred iQGME on the Ionosphere data with 25% features missing in terms of (a) performance, (b) time consumed for each iteration, (c) number of iterations. For the VB inference, we set a threshold (10^{-6}) for the relative change of lower bound in two consecutive iterations as the convergence criterion; for the MCMC inference, we discard the initial samples from the first 1000 iterations (burn-in), and collect the next 500 samples to present the posterior.

6.3 Unexploded Ordnance Data

We now consider an unexploded ordnance (UXO) detection problem (Zhang et al., 2003), where two types of sensors are used to collect data, but one of them may be absent for particular targets. Specifically, one sensor is a magnetometer (MAG) and the other an electromagnetic induction (EMI) sensor; these sensors are deployed separately to interrogate buried targets, and for some targets both sensors are deployed and for others only one sensor is deployed. This is a real sensing problem for which missing data occurs naturally. The data considered were made available to the authors by the US Army (and were collected from a real former bombing range in the US); the data are available to other researchers upon request. The total number of targets are 146, where 79 of them UXO and the rest are non-UXO (i.e., non-explosives). A six-dimensional feature vector is extracted from the raw signals to represent each target, with the first three components corresponding to MAG features and the rest as EMI features (details on feature extraction is provided in Zhang et al., 2003). Figure 12 shows the missing patterns for this data set.

We compare the proposed iQGME-VB algorithm with the SVM, RVM and LR-Integration as detailed in Section 6.2. In order to evaluate the overall performance of classifiers, we randomly partition the training and test subsets, and change the training size. Results are shown in Figure 13, where only performance means are reported for the legibility of the figures. From this figure, the proposed iQGME-VB method is robust for all the experimental settings under both performance criteria.

6.4 Sepsis Classification Data

In Sections 6.2 and 6.3, we have demonstrated the proposed iQGME-VB on data sets with low to moderate dimensionality. A high-dimensional data set with natural missing values is considered in this subsection. These data were made available to the authors from the National Center for Genomic Research in the US, and will be made available upon request. This is another example

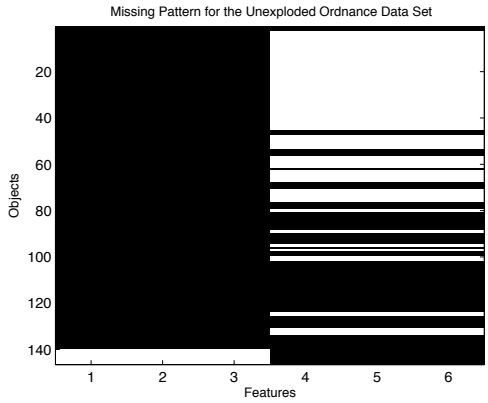
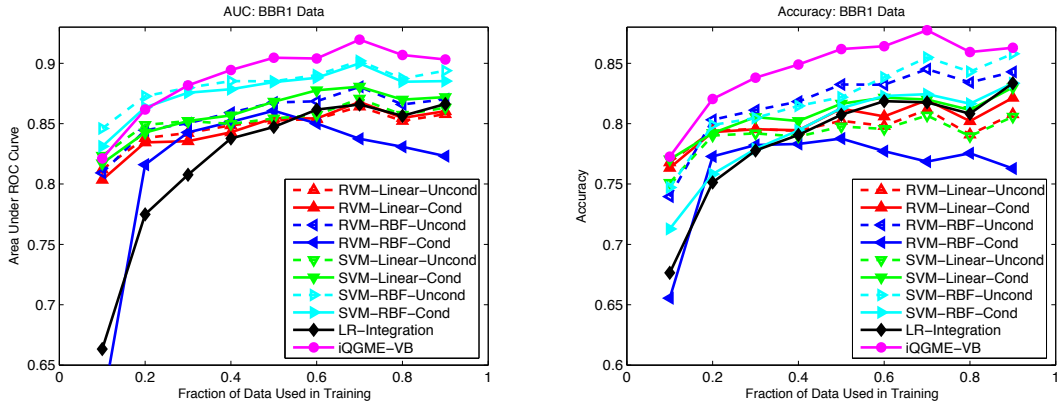


Figure 12: Missing pattern for the unexploded ordnance data set, where black and white indicate observed and missing, respectively.



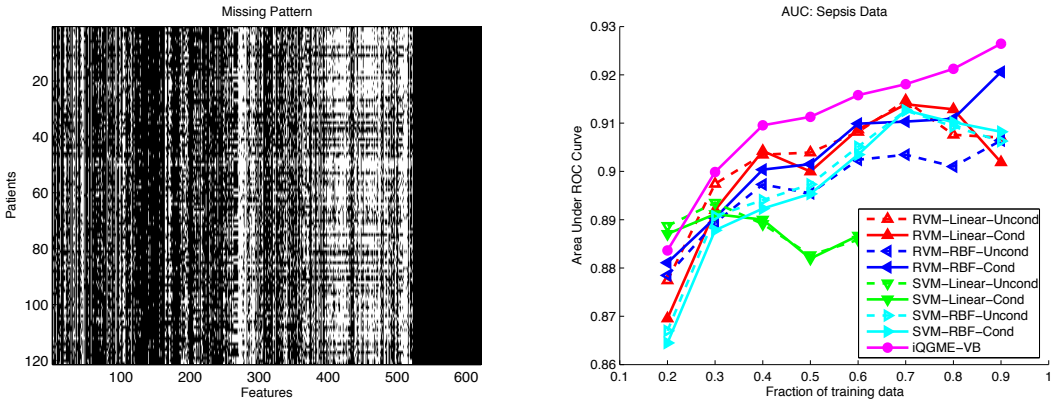
(a) (b)

Figure 13: Mean performance over 100 random training/test partitions for each training fraction on the unexploded ordnance data set, in terms of (a) area under the ROC curve, and (b) classification accuracy.

for which missing data are a natural consequence of the sensing modality. There are 121 patients who are infected by sepsis, with 90 of them surviving (label -1) and 31 of them who die (label 1). For each patient, we have 521 metabolic features and 100 protein features. The purpose is to predict whether a patient infected by sepsis will die given his/her features. The missing pattern of feature values is shown in Figure 14(a), where black indicates observed (this missingness is a natural consequence of the sensing device).

As the data are in a 621-dimensional feature space, with only 121 samples available, we use the MFA-based variant of the iQGME (Section 2.4). To impose the low-rank belief for each cluster, we set $c_0 = d_0 = 1$, and the largest possible dimensionality for clusters is set to be $L = 50$.

We compare to the same algorithms considered in Section 6.3, except the LR-Integration algorithm since it is not capable of handling such a high-dimensional data set. Mean AUC over ten random partitions are reported in Figure 14(b). Here we report the SVM and RVM results on the original data since they are able to classify the data in the original 621-dimensional space after missing values are imputed; we also examined SVM and RVM results on the data in a lower-dimensional latent space, after first performing factor analysis on the data, and these results were very similar to the SVM/RVM results in the original 621-dimensional space. From Figure 14(b), our method provides improvement by handling missing values analytically in the procedure of model inference and performing a dimensionality reduction jointly with local classifiers learning.



(a) (b)

Figure 14: Sepsis data set. (a) Missing pattern, where black and white indicate observed and missing, respectively, (b) mean performance over 100 random training/test partitions for each training fraction.

6.5 Multi-Task Learning with Landmine Detection Data

We now consider a multi-task-learning example. In an application of landmine detection (available at `http://www.ee.duke.edu/~lcarin/LandmineData.zip`), data collected from 19 landmine fields are treated as 19 subtasks. Among them, subtasks 1-10 correspond to regions that are relatively highly foliated and subtasks 11-19 correspond to regions that are bare earth or desert. In all subtasks, each target is characterized by a 9-dimensional feature vector $\boldsymbol{x}$ with corresponding binary label y (1 for landmines and -1 for clutter). The number of landmines and clutter in each task is summarized in Figure 15. The feature vectors are extracted from images measured with airborne radar systems. A detailed description of this landmine data set has been presented elsewhere (Xue et al., 2007).

Although our main objective is to simultaneously learn classifiers for multiple tasks with incomplete data, we first demonstrate the proposed iQGME-based multi-task learning (MTL) model on the complete data, comparing it to two multi-task learning algorithms designed for the situation with all the features observed. One is based on task-specific logistic regression (LR) models, with the DP as a hierarchical prior across all the tasks (Xue et al., 2007); the other assumes an underlying

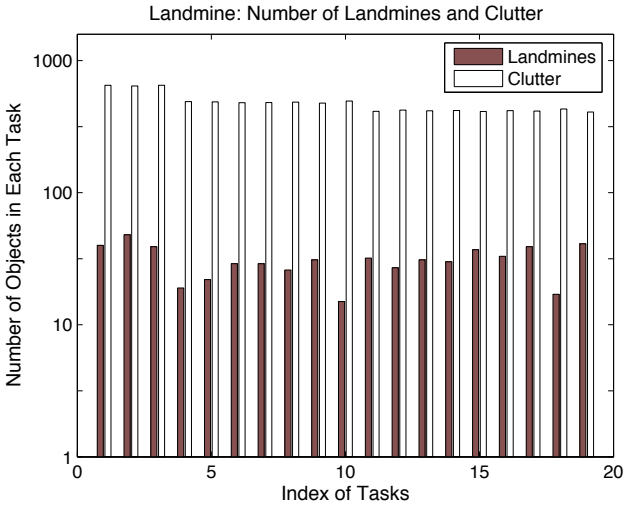


Figure 15: Number of landmines and clutter in each task for the landmine-detection data set (Xue et al., 2007).

structure, which is shared by all the tasks (Ando and Zhang, 2005). For the LR-MTL algorithm, we cite results on complete data from Xue et al. (2007), and implement the authors' Matlab code with default hyper-parameters on the cases with incomplete data. The Matlab implementation for the Structure-MTL algorithm is included in the "Transfer Learning Toolkit for Matlab" available at `http://multitask.cs.berkeley.edu/`. The dimension of the underlying structure is a user-set parameter, and it should be smaller than the feature dimension in the original space. As the dimension of the landmine detection data is 9, we set the hidden dimension as 5. We also tried 6, 7, and 8, and did not observe big differences in performance. Single-task learning (STL) iQGME and LR models are also included for comparison.

Each task is divided into training and test subsets randomly. Since the number of elements in the two classes is highly unbalanced, as shown in Figure 15, we impose that there is at least one instance from each class in each subset. Following Xue et al. (2007), the size of the training subset in each task varies from 20 to 300 in increments of 20, and 100 independent trials are performed for each size of data set. An average AUC (Hanley and McNeil, 1982) over all the 19 tasks is calculated as the performance representation for one trial of a given training size. Results are reported in Figure 16.

The first observation from Figure 16 is that we obtain a significant performance improvement for single-task learning by using the iQGME-VB instead of the linear logistic regression model (Xue et al., 2007). We also notice that the multi-task algorithm based on iQGME-VB further improves the performance when the training data are scarce, and yields comparable overall results as the LR-MTL does. The structure-MTL does not perform well on this data set. We suspect that a hidden structure in such a 9-dimensional space does not necessarily exist. Another possible reason may be that the minimization of empirical risk is sensitive for the cases with highly unbalanced labels, as for this data set.

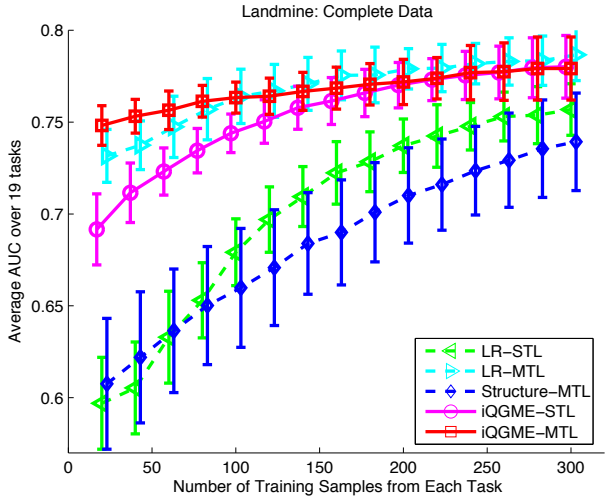
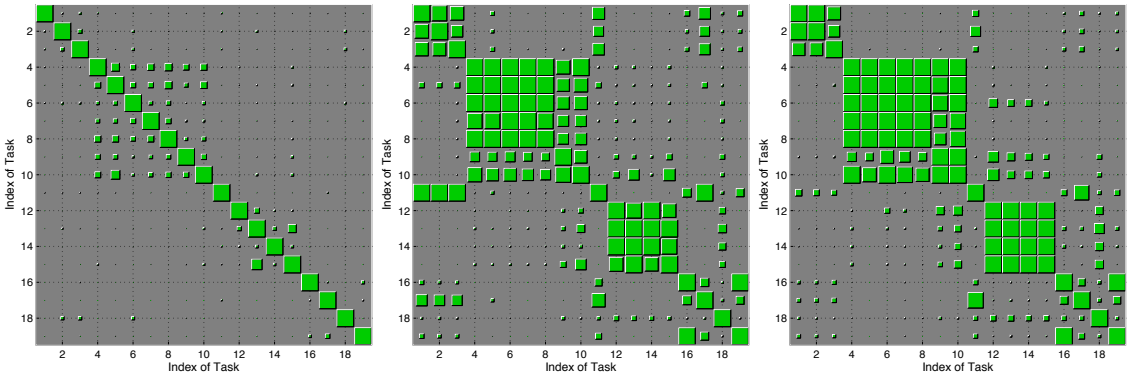


Figure 16: Average AUC over 19 tasks of landmine detection with complete data. Error bars reflect the standard deviation across 100 random partitions of training and test subsets. Results of logistic regression based algorithms are cited from Xue et al. (2007), where LR-MTL and LR-STL respectively correspond to SMTL-2 and STL in Figure 3 of Xue et al. (2007).

It is also interesting to explore the similarity among tasks. The similarity defined by different algorithms may be different. In Xue et al. (2007), two tasks are defined to be similar if they share the same linear classifier. However, with the joint distribution of covariates and the response, the iQGME-MTL requires both the data distributions and the classification boundaries to be similar if two tasks are deemed to be similar. Another difference is that two tasks could be partially similar since sharing between tasks is encouraged at the cluster-level instead of at the task-level (Xue et al. 2007 employs task-level clustering). We generate the similarity matrices between tasks as follows: In each random trial, there are in total S higher-level items shared among tasks. For each task, we can find the task-specific probability mass function (pmf) over all the higher-level items. Using these pmfs as the characteristics for tasks in the current trial, we calculate the pair-wise Kullback-Leibler (KL) distances and convert them to similarity measures through a minus exponential function. Results of multiple trials are summed over and normalized as shown in Figure 17. It can be seen that the similarity structure among tasks becomes clearer when we have more training data available. As discovered by Xue et al. (2007), we also find two big clusters correspond to two different vegetation conditions of the landmine fields (task 1-10 and task 11-19). Further sub-structures among tasks are also explored by the iQGME-MTL model, which may suggest other unknown difference among the landmine fields.

After yielding competitive results on the landmine-detection data set with complete data, the iQGME-based algorithms are evaluated on incomplete data, which are simulated by randomly removing a portion of feature values for each task as in Section 6.2. We consider three different



(a) (b) (c)

Figure 17: Similarity between tasks in the landmine detection problem with complete data given (a) 20, (b) 100, and (c) 300 training samples from each task. The size of green blocks represent the value of the corresponding matrix element.

portions of missing values: 25%, 50% and 75%. As in the experiments above on benchmark data sets, we perform ten independent random trials for each setting of missing fraction and training size.

To the best of our knowledge, there exists no previous work in the literature on multi-task learning with missing data. As presented in Figure 18, we use the LR-MTL (Xue et al., 2007) and the Structure-MTL (Ando and Zhang, 2005) with missing values imputed as baseline algorithms. Results of the two-step LR with integration (Williams et al., 2007) and the LR-STL with single imputations are also included for comparison. Imputations using both unconditional-means and conditional-means are considered. From Figure 18, iQGME-STL consistently performs best among single-task learning methods and even better than LR-MTL-Uncond when the size of the training set is relatively large. The imputations using conditional-means yields consistently better results for the LR-based models on this data set. The iQGME-MTL outperforms the baselines and all the single-task learning methods overall. Furthermore, the improvement of iQGME-MTL is more pronounced when there are more features missing. These observations underscore the advantage of handling missing data in a principled manner and at the same time learning multiple tasks simultaneously.

The task-similarity matrices for the incomplete-data cases are shown in Figure 19. It can be seen that when a small fraction (e.g., 25%) of the feature values are missing and training data are rich (e.g., 300 samples from each task), the similarity pattern among tasks is similar to what we have seen for the complete-data case. As the fraction of missing values becomes larger, tasks appear more different from each other in terms of the usage of the higher-level items. Considering that the missing pattern for each task is unique, it is probable that tasks look quite different from each other after a large fraction of feature values are missing. However, the fact that tasks tend to use different subsets of higher-level items does not mean it is equivalent to learning them separately (STL), as parameters of the common base measures are inferred based on all the tasks.

6.6 Multi-Task Learning with Handwritten Letters Data

The final example corresponds to multi-task learning of classifiers for handwritten letters, this data set included in the "Transfer Learning Toolkit for Matlab" available at `http://multitask.cs.`

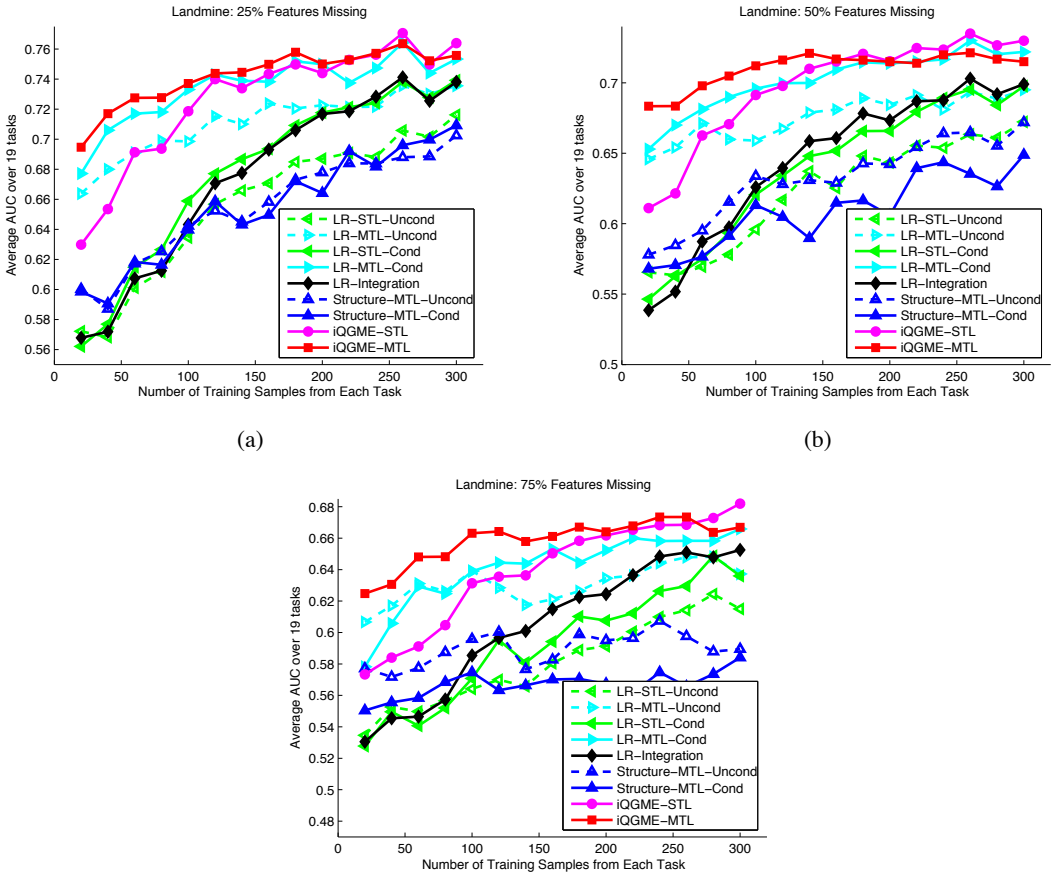


Figure 18: Average AUC over 19 tasks of landmine detection for the cases when (a) 25%, (b) 50%, and (c) 75% of the features are missing. Mean values of performance across 10 random partitions of training and test subsets are reported. Error bars are omitted for legibility.

`berkeley.edu/`. The objective of each task is to distinguish two letters which are easily confused. The number of samples for all the letters considered in the total eight tasks is summarized in Table 2. Each sample is a 16×8 image as shown in Figure 20. We use the 128 pixel values of each sample directly as its feature vector.

Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
'c': 2107	'g': 2460	'm': 1596	'a': 4016	'i': 4895	'a': 4016	'f': 918	'h': 858
'e': 4928	'y': 1218	'n': 5004	'g': 2460	'j': 188	'o': 3880	't': 2131	'n': 5004

Table 2: Handwritten letters classification data set.

We compare the proposed iQGME-MTL algorithm to the LR-MTL (Xue et al., 2007) and the Structure-MTL (Ando and Zhang, 2005) mentioned in Section 6.5. For the non-parametric Bayesian methods (iQGME-MTL and LR-MTL), we use the same parameter setting as before. The dimension

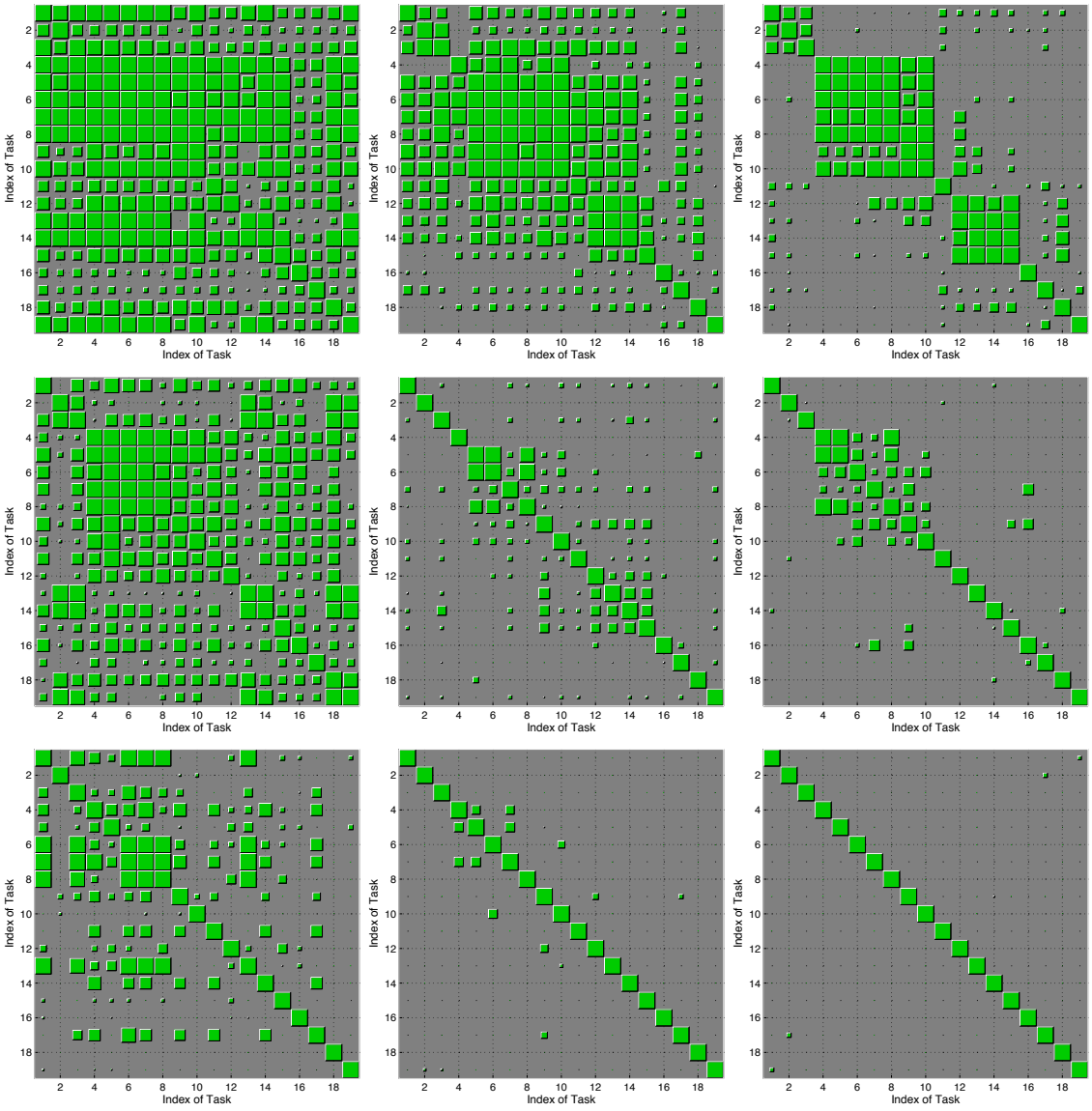


Figure 19: Similarity between tasks in the landmine detection problem with incomplete data. Row 1, 2 and 3 corresponds to the cases with 25%, 50% and 75% features missing, respectively; column 1, 2 and 3 corresponds to the cases with 20, 100 and 300 training samples from each task, respectively.

of the underlying structure for the Structure-MTL is set to be 50 in the results shown in Figure 21. We also tried 10, 20, 40, 60, 80 and 100, and did not observe big difference. From Figure 21, the iQGME-MTL performs significantly better than the baselines on this data set for all the missing fractions and training fractions under consideration. As we expected, the Structure-MTL yields comparable results as the LR-MTL on this data set.

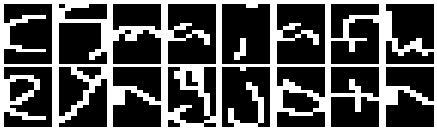


Figure 20: Sample images of the handwritten letters. The two images in each column represents the two classes in the corresponding task described in Table 2.

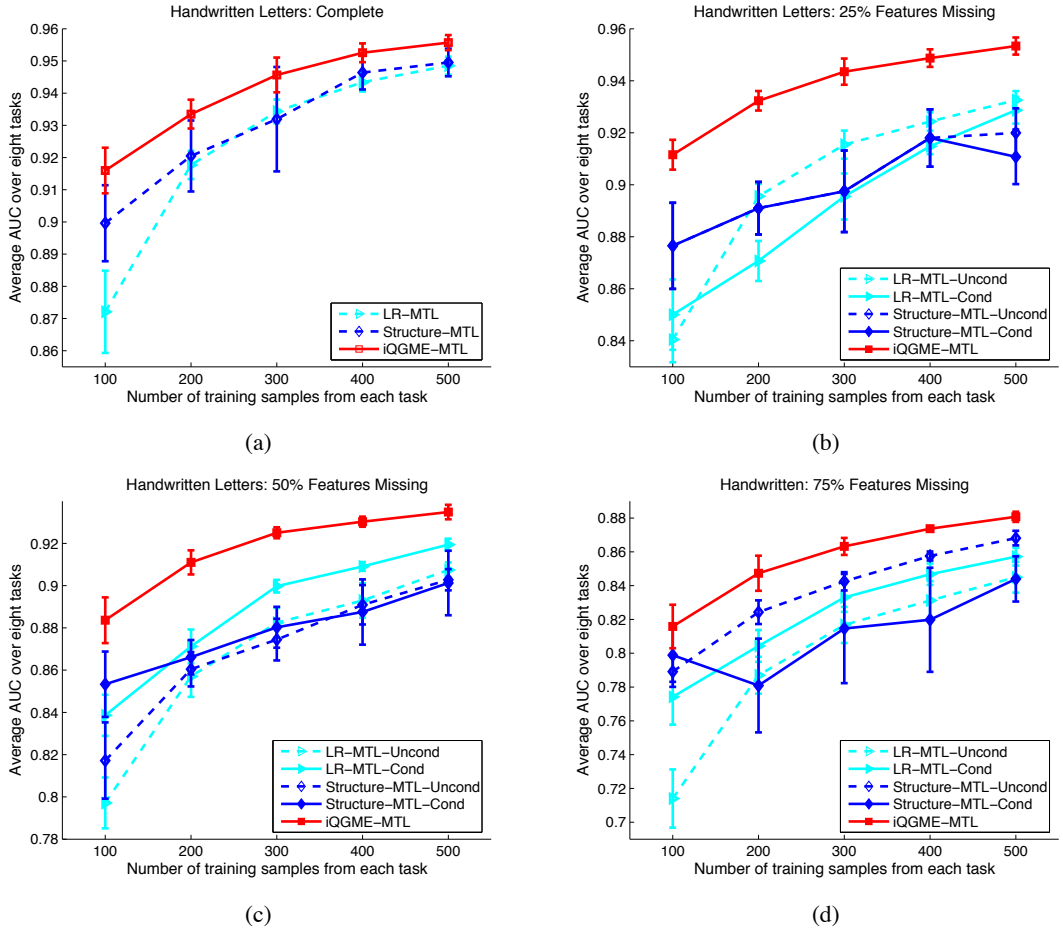


Figure 21: Average AUC over eight tasks of handwriting letters classification for the cases when (a) none, (b) 25%, (c) 50%, and (d) 75% of the features are missing. Mean values of performance with one standard deviation across 10 random partitions of training and test subsets are reported.

7. Conclusion and Future Work

In this paper we have introduced three new concepts, summarized as follows. First, we have employed non-parametric Bayesian techniques to develop a mixture-of-experts algorithm for classifier

design, which employs a set of localized (in feature space) linear classifiers as experts. The Dirichlet process is employed to allow the model to infer automatically the proper number of experts and their characteristics; in fact, since a Bayesian formulation is employed, a full posterior distribution is manifested on the properties of the local experts, including their number. Secondly, the classifier is endowed with the ability to naturally address missing data, without the need for an imputation step. Finally, the whole framework has been placed within the context of a multi-task learning, allowing one to jointly infer classifiers for multiple data sets with missing data. The multi-task-learning component has also been implemented with the general tools associated with the Dirichlet process, with specific implementations manifested via the hierarchical Dirichlet process. Because of the hierarchical form of the model, in terms of a sequence of distributions in the conjugate-exponential family, all inference has been manifested efficiently via variational Bayesian (VB) analysis. The VB results have been compared to those computed via Gibbs sampling; the VB results have been found to be consistent with those inferred via Gibbs sampling, while requiring a small fraction of the computational costs. Results have been presented for single-task and multi-task learning on various data sets with the same hyper-parameters setting (no model-parameter tuning), and encouraging algorithm performance has been demonstrated.

Concerning future research, we note that the use of multi-task learning provides an important class of contextual information, and therefore is particularly useful when one has limited labeled data and when the data are incomplete (missing features). Another form of context that has received significant recent attention is semi-supervised learning (Zhu, 2005). There has been recent work on integrating multi-task learning with semi-supervised learning (Liu et al., 2007). An important new research direction includes extending semi-supervised multi-task learning to realistic problems for which the data are incomplete.

Appendix A.

The update equations of single-task learning with incomplete data are summarized as follows:

1. $q(t_i|\mu_i^t)$

$$\mu_i^t = \sum_{h=1}^{N} \rho_{ih} \langle \boldsymbol{w}_h \rangle^T \hat{\boldsymbol{x}}^b{}_{i,h} \quad \text{where} \quad \hat{\boldsymbol{x}}^b{}_{i,h} = [\boldsymbol{x}_i^{o_i}; \boldsymbol{m}_h^{m_i|o_i}; 1].$$

   The expectation of t_i and t_i^2 may be derived according to properties of truncated normal distributions:

$$\begin{aligned} \langle t_i \rangle &= \mu_i^t + \frac{\phi(-\mu_i^t)}{\mathbf{1}(y_i = 1) - \Phi(-\mu_i^t)}, \\ \langle t_i^2 \rangle &= 1 + (\mu_i^t)^2 + \frac{\mu_i^t \phi(-\mu_i^t)}{\mathbf{1}(y_i = 1) - \Phi(-\mu_i^t)}, \end{aligned}$$

   where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the probability density function and the cumulative density function of the standard normal distribution, respectively.

2. $q(\boldsymbol{x}_i^{m_i}, z_i | \boldsymbol{m}_{ih}^{m_i|o_i}, \boldsymbol{\Sigma}_{ih}^{m_i|o_i}, \rho_{ih})$
   A related derivation for a GMM model with incomplete data could be found in Williams et al. (2007), where no classifier terms appear.

First, we explicitly write the intercept w_h^b , that is, $\boldsymbol{w}_h = [(\boldsymbol{w}_h^x)^T, w_h^b]^T$:

$$\begin{aligned} & q(\boldsymbol{x}_i^{m_i}, z_i = h) \\ \propto\ & \exp\{\langle \ln[p(t_i|z_i = h, \boldsymbol{x}_i, \boldsymbol{W}) p(z_i = h|V) p(\boldsymbol{x}_i|z_i = h, \boldsymbol{\mu}, \boldsymbol{\Lambda})]\rangle_{q(t_i)q(\boldsymbol{w}_h)q(V)q(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h)}\} \\ \propto\ & A_{ih} \mathcal{N}_P(\boldsymbol{x}_i|\tilde{\boldsymbol{\mu}}_{ih}, \tilde{\boldsymbol{\Sigma}}_{ih}), \end{aligned}$$

where

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}}_{ih} &= [\langle \boldsymbol{w}_h^x (\boldsymbol{w}_h^x)^T \rangle + \nu_h \boldsymbol{B}_h]^{-1} \\ \tilde{\boldsymbol{\mu}}_{ih} &= \tilde{\boldsymbol{\Sigma}}_{ih}[\langle t_i \rangle \langle \boldsymbol{w}_h^x \rangle + \nu_h \boldsymbol{B}_h \boldsymbol{m}_h - \langle \boldsymbol{w}_h^x w_h^b \rangle] \\ A_{ih} &= \exp\{\langle \ln V_h \rangle + \sum_{l<h} \langle \ln(1 - V_l) \rangle + \langle t_i \rangle \langle w_h^b \rangle \\ &\quad + \frac{1}{2}[\langle \ln|\boldsymbol{\Lambda}_h| \rangle + \tilde{\boldsymbol{\mu}}_{ih}^T \tilde{\boldsymbol{\Sigma}}_{ih}^{-1} \tilde{\boldsymbol{\mu}}_{ih} + \ln|\tilde{\boldsymbol{\Sigma}}_{ih}| - \frac{P}{u_h} - \boldsymbol{m}_h^T \nu_h \boldsymbol{B}_h \boldsymbol{m}_h - \langle (w_h^b)^2 \rangle]\}. \end{aligned}$$

Since

$$\begin{bmatrix} \boldsymbol{x}_i^{o_i} \\ \boldsymbol{x}_i^{m_i} \end{bmatrix} \sim \mathcal{N}_P \left(\begin{bmatrix} \tilde{\boldsymbol{\mu}}_{ih}^{o_i} \\ \tilde{\boldsymbol{\mu}}_{ih}^{m_i} \end{bmatrix}, \begin{bmatrix} \tilde{\boldsymbol{\Sigma}}_{ih}^{o_i o_i} & \tilde{\boldsymbol{\Sigma}}_{ih}^{o_i m_i} \\ \tilde{\boldsymbol{\Sigma}}_{ih}^{m_i o_i} & \tilde{\boldsymbol{\Sigma}}_{ih}^{m_i m_i} \end{bmatrix} \right),$$

the conditional distribution of missing features $\boldsymbol{x}_i^{m_i}$ given observable features $\boldsymbol{x}_i^{o_i}$ is also a normal distribution, that is, $\boldsymbol{x}_i^{m_i}|\boldsymbol{x}_i^{o_i} \sim \mathcal{N}_{|m_i|}(\boldsymbol{m}_h^{m_i|o_i}, \boldsymbol{\Sigma}_h^{m_i|o_i})$ with

$$\begin{aligned} \boldsymbol{m}_h^{m_i|o_i} &= \tilde{\boldsymbol{\mu}}_{ih}^{m_i} + \tilde{\boldsymbol{\Sigma}}_{ih}^{m_i o_i} (\tilde{\boldsymbol{\Sigma}}_{ih}^{o_i o_i})^{-1} (\boldsymbol{x}_i^{o_i} - \tilde{\boldsymbol{\mu}}_{ih}^{o_i}), \\ \boldsymbol{\Sigma}_h^{m_i|o_i} &= \tilde{\boldsymbol{\Sigma}}_{ih}^{m_i m_i} - \tilde{\boldsymbol{\Sigma}}_{ih}^{m_i o_i} (\tilde{\boldsymbol{\Sigma}}_{ih}^{o_i o_i})^{-1} \tilde{\boldsymbol{\Sigma}}_{ih}^{o_i m_i}. \end{aligned}$$

Therefore, $q(\boldsymbol{x}_i^{m_i}, z_i = h)$ could be factorized as the product of a factor independent of $\boldsymbol{x}_i^{m_i}$ and the variational posterior of $\boldsymbol{x}_i^{m_i}$, that is,

$$\begin{aligned} q(\boldsymbol{x}_i^{m_i}, z_i = h) &\propto A_{ih} \mathcal{N}_{|o_i|}(\boldsymbol{x}_i^{o_i}|\tilde{\boldsymbol{\mu}}_{ih}^{o_i}, \tilde{\boldsymbol{\Sigma}}_{ih}^{o_i o_i}) \mathcal{N}_{|m_i|}(\boldsymbol{x}_i^{m_i}|\boldsymbol{m}_h^{m_i|o_i}, \boldsymbol{\Sigma}_h^{m_i|o_i}) \\ \rho_{ih} &\propto A_{ih} \mathcal{N}_{|o_i|}(\boldsymbol{x}_i^{o_i}|\tilde{\boldsymbol{\mu}}_{ih}^{o_i}, \tilde{\boldsymbol{\Sigma}}_{ih}^{o_i o_i}) \end{aligned}$$

For complete data, no factorization for the distribution for $\boldsymbol{x}_i^{m_i}$ is necessary:

$$\begin{aligned} \rho_{ih} \propto\ & \exp\{\langle t_i \rangle \langle \boldsymbol{w}_h \rangle^T \boldsymbol{x}_i - \frac{1}{2} \boldsymbol{x}_i^T \langle \boldsymbol{w}_h \boldsymbol{w}_h^T \rangle \boldsymbol{x}_i + \langle \ln V_h \rangle \\ & + \sum_{l<h} \langle \ln(1 - V_l) \rangle + \frac{1}{2} \langle \ln|\boldsymbol{\Lambda}_h| \rangle - \frac{1}{2} \langle (\boldsymbol{x}_i - \boldsymbol{\mu}_h)^T \boldsymbol{\Lambda}_h (\boldsymbol{x}_i - \boldsymbol{\mu}_h) \rangle\} \end{aligned}$$

3. $q(V_h|v_{h1}, v_{h2})$

Similar updating could be found in Blei and Jordan (2006), except that we put a prior belief on α here instead of setting a fixed number.

$$\begin{aligned} & v_{h1} = 1 + \sum_{i=1}^n \rho_{ih}, \quad v_{h2} = \langle \alpha \rangle + \sum_{i=1}^n \sum_{l>h} \rho_{il}; \\ & \langle \ln V_h \rangle = \psi(v_{h1}) - \psi(v_{h1} + v_{h2}), \quad \langle \ln(1 - V_l) \rangle = \psi(v_{l2}) - \psi(v_{l1} + v_{l2}). \end{aligned}$$

4. $q(\boldsymbol{\mu}_h, \boldsymbol{\Lambda}_h | \boldsymbol{m}_h, u_h, \boldsymbol{B}_h, \nu_h)$

   Similar updating could be found in Williams et al. (2007).

$$\nu_h = \nu_0 + N_h, \quad u_h = u_0 + N_h, \quad \boldsymbol{m}_h = \frac{u_0 \boldsymbol{m}_0 + N_h \bar{\boldsymbol{x}}_h}{u_h},$$
$$\boldsymbol{B}_h^{-1} = \boldsymbol{B}_0^{-1} + \sum_{i=1}^{n} \rho_{ih} \hat{\boldsymbol{\Omega}}_{i,h} + N_h \bar{\boldsymbol{S}}_h + \frac{u_0 N_h}{u_h} (\bar{\boldsymbol{x}}_h - \boldsymbol{m}_0)(\bar{\boldsymbol{x}}_h - \boldsymbol{m}_0)^T,$$

   where

$$N_h = \sum_{i=1}^{n} \rho_{ih}, \quad \bar{\boldsymbol{x}}_h = \sum_{i=1}^{n} \rho_{ih} \boldsymbol{x}_i / N_h, \quad \bar{\boldsymbol{S}}_h = \sum_{i=1}^{n} \rho_{ih} (\hat{\boldsymbol{x}}_{i,h} - \bar{\boldsymbol{x}}_h)(\hat{\boldsymbol{x}}_{i,h} - \bar{\boldsymbol{x}}_h)^T / N_h.$$
$$\hat{\boldsymbol{x}}_{i,h} = \begin{bmatrix} \boldsymbol{x}_i^{o_i} \\ \boldsymbol{m}_h^{m_i|o_i} \end{bmatrix}, \quad \hat{\boldsymbol{\Omega}}_{i,h} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_h^{m_i|o_i} \end{bmatrix}.$$

$$\langle \ln |\boldsymbol{\Lambda}_h| \rangle = \sum_{p=1}^{P} \psi((\nu_h - p + 1)/2) + P \ln 2 + \ln |\boldsymbol{B}_h|,$$

$$\langle (\boldsymbol{x}_i - \boldsymbol{\mu}_h)^T \boldsymbol{\Lambda}_h (\boldsymbol{x}_i - \boldsymbol{\mu}_h) \rangle = (\hat{\boldsymbol{x}}_{i,h} - \boldsymbol{m}_h)^T \nu_h \boldsymbol{B}_h (\hat{\boldsymbol{x}}_{i,h} - \boldsymbol{m}_h) + P/u_h + \mathrm{tr}(\nu_h \boldsymbol{B}_h \hat{\boldsymbol{\Omega}}_{i,h}).$$

5. $q(\boldsymbol{w}_h | \boldsymbol{\mu}_h^w, \boldsymbol{\Sigma}_h^w)$, $\langle \boldsymbol{w}_h \rangle = \boldsymbol{\mu}_h^w$, $\langle \boldsymbol{w}_h \boldsymbol{w}_h^T \rangle = \boldsymbol{\Sigma}_h^w + \boldsymbol{\mu}_h^w (\boldsymbol{\mu}_h^w)^T$.

$$\boldsymbol{\Sigma}_h^w = \left(\sum_{i=1}^{n} \rho_{ih} (\hat{\boldsymbol{\Omega}}_{i,h} + \hat{\boldsymbol{x}^b}_{i,h} \hat{\boldsymbol{x}^b}_{i,h}^T) + \mathrm{diag}(\langle \boldsymbol{\lambda} \rangle) \right)^{-1},$$
$$\boldsymbol{\mu}_h^w = \boldsymbol{\Sigma}_h^w \left(\sum_{i=1}^{n} \rho_{ih} \hat{\boldsymbol{x}^b}_{i,h} \langle t_i \rangle + \mathrm{diag}(\langle \boldsymbol{\lambda} \rangle) \boldsymbol{\phi} \right).$$

6. $q(\zeta_p, \lambda_p | \phi_p, \gamma, a_p, b_p)$, $\langle \lambda_p \rangle = a_p / b_p$.

   Similar updating could be found in Xue et al. (2007).

$$\phi_p = \sum_{h=1}^{N} \langle w_{hp} \rangle / \gamma, \quad \gamma = \gamma_0 + N$$
$$a_p = a_0 + \frac{N}{2}, \quad b_p = b_0 + \frac{1}{2} \sum_{h=1}^{N} \langle w_{hp}^2 \rangle - \frac{1}{2} \gamma \phi_p^2.$$

7. $q(\alpha | \tau_1, \tau_2)$, $\langle \alpha \rangle = \tau_1 / \tau_2$.

   Similar updating could be found in any VB-inferred DP model with a Gamma prior on α (Xue et al., 2007).

$$\tau_1 = N - 1 + \tau_{10}, \quad \tau_2 = \tau_{20} - \sum_{h=1}^{N-1} \langle \ln(1 - V_h) \rangle.$$

The update equations of multi-task learning with incomplete data are summarized as follows:

1. $q(t_{ji}|\mu^t_{ji})$

$$\mu^t_{ji} = \sum_{s=1}^{S} E\sigma_{jis}\langle \boldsymbol{w}_s\rangle^T \hat{\boldsymbol{x}^b}_{ji} \quad \text{where} \quad E\sigma_{jis} = \sum_{h=1}^{N} \rho_{jih}\sigma_{jhs}, \quad \hat{\boldsymbol{x}^b}_{ji} = [\boldsymbol{x}^{o_{ji}}_{ji}; \boldsymbol{m}^{m_{ji}|o_{ji}}_{ji}; 1].$$

2. $q(x^{m_{ji}}_{ji}|\boldsymbol{m}^{m_{ji}|o_{ji}}_{ji}, \boldsymbol{\Sigma}^{m_{ji}|o_{ji}}_{ji})$

$$\begin{aligned} \boldsymbol{m}^{m_{ji}|o_{ji}}_{ji} &= \tilde{\boldsymbol{\mu}}^{m_{ji}}_{ji} + \tilde{\boldsymbol{\Sigma}}^{m_{ji}o_{ji}}_{ji}(\tilde{\boldsymbol{\Sigma}}^{o_{ji}o_{ji}}_{ji})^{-1}(\boldsymbol{x}^{o_{ji}}_{ji} - \tilde{\boldsymbol{\mu}}^{o_{ji}}_{ji}), \\ \boldsymbol{\Sigma}^{m_{ji}|o_{ji}}_{ji} &= \tilde{\boldsymbol{\Sigma}}^{m_{ji}m_{ji}}_{ji} - \tilde{\boldsymbol{\Sigma}}^{m_{ji}o_{ji}}_{ji}(\tilde{\boldsymbol{\Sigma}}^{o_{ji}o_{ji}}_{ji})^{-1}\tilde{\boldsymbol{\Sigma}}^{o_{ji}m_{ji}}_{ji}, \end{aligned}$$

where

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}}_{ji} &= \left(\sum_{s=1}^{S} E\sigma_{jis}(\langle \boldsymbol{w}^x_s(\boldsymbol{w}^x_s)^T\rangle + \nu_s \boldsymbol{B}_s)\right)^{-1}, \\ \tilde{\boldsymbol{\mu}}_{ji} &= \tilde{\boldsymbol{\Sigma}}_{ji}\sum_{s=1}^{S} E\sigma_{jis}(\langle t_{ji}\rangle\langle \boldsymbol{w}^x_s\rangle + \nu_s \boldsymbol{B}_s\boldsymbol{m}_s - \langle \boldsymbol{w}^x_s w^b_s\rangle). \end{aligned}$$

$$\hat{\boldsymbol{x}}_{ji} = \begin{bmatrix} \boldsymbol{x}^{o_{ji}}_{ji} \\ \boldsymbol{m}^{m_{ji}|o_{ji}}_{ji} \end{bmatrix}, \quad \hat{\boldsymbol{\Omega}}_{ji} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}^{m_{ji}|o_{ji}}_{ji} \end{bmatrix}, \quad \langle \boldsymbol{x}_{ji}\boldsymbol{x}^T_{ji}\rangle = \hat{\boldsymbol{x}}_{ji}\hat{\boldsymbol{x}}^T_{ji} + \hat{\boldsymbol{\Omega}}_{ji}.$$

3. $q(z_{ji}|\boldsymbol{\rho}_{ji})$

$$\begin{aligned} & \rho_{jih} = q(z_{ji} = h) \\ \propto\ & \exp\{\sum_{s=1}^{S}\sigma_{jhs}[\langle t_{ji}\rangle\langle \boldsymbol{w}_s\rangle^T \hat{\boldsymbol{x}^b}_{ji} - \frac{1}{2}\mathrm{tr}(\langle \boldsymbol{w}_s\boldsymbol{w}^T_s\rangle\langle \boldsymbol{x}^b_{ji}(\boldsymbol{x}^b_{ji})^T\rangle)] \\ & + \langle \ln V_{jh}\rangle + \sum_{l<h}\langle \ln(1-V_{jl})\rangle \\ & + \frac{1}{2}\sum_{s=1}^{S}\sigma_{jhs}[\langle \ln|\boldsymbol{\Lambda}_s|\rangle - (\hat{\boldsymbol{x}}_{ji} - \boldsymbol{m}_s)^T\nu_s\boldsymbol{B}_s(\hat{\boldsymbol{x}}_{ji} - \boldsymbol{m}_s) - P/u_s - \mathrm{tr}(\nu_s\boldsymbol{B}_s\hat{\boldsymbol{\Omega}}_{ji})]\}. \end{aligned}$$

4. $q(\boldsymbol{V}|\boldsymbol{v})$

$$v_{jh1} = 1 + \sum_{i=1}^{n_j}\rho_{jih}, \qquad v_{jh2} = \langle\alpha\rangle + \sum_{l>h}\sum_{i=1}^{n_j}\rho_{jil}.$$

$$\langle \ln V_{jh}\rangle = \psi(v_{jh1}) - \psi(v_{jh1} + v_{jh2}), \qquad \langle \ln(1-V_{jl})\rangle = \psi(v_{jl2}) - \psi(v_{jl1} + v_{jl2}).$$

5. $q(\alpha|\tau_1,\tau_2)$, $\langle\alpha\rangle = \tau_1/\tau_2$.

$$\tau_1 = J(N-1) + \tau_{10}, \quad \tau_2 = \tau_{20} - \sum_{j=1}^{J}\sum_{h=1}^{N-1}\langle \ln(1-V_{jh})\rangle.$$

6. $q(\boldsymbol{c}|\boldsymbol{\sigma})$

$$\begin{aligned}\sigma_{jhs} \quad \propto \quad & \exp\{\sum_{i=1}^{n_j}\rho_{jih}[\langle t_{ji}\rangle\langle \boldsymbol{w}_s\rangle^T\hat{\boldsymbol{x}^b}_{ji}-\frac{1}{2}\mathrm{tr}(\langle \boldsymbol{w}_s\boldsymbol{w}_s^T\rangle\langle \boldsymbol{x}^b_{ji}(\boldsymbol{x}^b_{ji})^T\rangle)] \\ & +\langle\ln U_s\rangle+\sum_{l<s}\langle\ln(1-U_s)\rangle \\ & +\frac{1}{2}\sum_{i=1}^{n_j}\rho_{jih}[\langle\ln|\boldsymbol{\Lambda}_s|\rangle-(\hat{\boldsymbol{x}}_{ji}-\boldsymbol{m}_s)^T\nu_s\boldsymbol{B}_s(\hat{\boldsymbol{x}}_{ji}-\boldsymbol{m}_s)-P/u_s-\mathrm{tr}(\nu_s\boldsymbol{B}_s\hat{\boldsymbol{\Omega}}_{ji})]\}.\end{aligned}$$

7. $q(U_s|\kappa_{s1},\kappa_{s2})$

$$\kappa_{s1}=1+\sum_{j=1}^{J}\sum_{h=1}^{N}\sigma_{jhs},\quad \kappa_{s2}=\langle\beta\rangle+\sum_{j=1}^{J}\sum_{h=1}^{N}\sum_{l>s}\sigma_{jhl}.$$
$$\langle\ln U_s\rangle=\psi(\kappa_{s1})-\psi(\kappa_{s1}+\kappa_{s2}),\quad \langle\ln(1-U_s)\rangle=\psi(\kappa_{s2})-\psi(\kappa_{s1}+\kappa_{s2}).$$

8. $q(\beta|\tau_3,\tau_4)$, $\langle\beta\rangle=\tau_3/\tau_4$.

$$\tau_3=S-1+\tau_{30},\quad \tau_4=\tau_{40}-\sum_{s=1}^{S-1}\langle\ln(1-U_s)\rangle.$$

9. $q(\boldsymbol{\mu}_s,\boldsymbol{\Lambda}_s|\boldsymbol{m}_s,u_s,\boldsymbol{B}_s,\nu_s)$

$$\nu_s=\nu_0+N_s,\quad u_s=u_0+N_s,\quad \boldsymbol{m}_s=\frac{u_0\boldsymbol{m}_0+N_s\bar{\boldsymbol{x}}_s}{u_s},$$
$$\boldsymbol{B}_s^{-1}=\boldsymbol{B}_0^{-1}+\sum_{j=1}^{J}\sum_{i=1}^{n_j}E\sigma_{jis}\hat{\boldsymbol{\Omega}}_{ji}+N_s\bar{\boldsymbol{S}}_s+\frac{u_0N_s}{u_s}(\bar{\boldsymbol{x}}_s-\boldsymbol{m}_0)(\bar{\boldsymbol{x}}_s-\boldsymbol{m}_0)^T,$$

where $E\sigma_{jis}=\sum_{h=1}^{N}\rho_{jih}\sigma_{jhs}$, and

$$N_s=\sum_{j=1}^{J}\sum_{i=1}^{n_j}E\sigma_{jis},\quad \bar{\boldsymbol{x}}_s=\sum_{j=1}^{J}\sum_{i=1}^{n_j}E\sigma_{jis}\hat{\boldsymbol{x}}_{ji}/N_s,$$
$$\bar{\boldsymbol{S}}_s=\sum_{j=1}^{J}\sum_{i=1}^{n_j}E\sigma_{jis}(\hat{\boldsymbol{x}}_{ji}-\bar{\boldsymbol{x}}_s)(\hat{\boldsymbol{x}}_{ji}-\bar{\boldsymbol{x}}_s)^T/N_s.$$

$$\langle\ln|\boldsymbol{\Lambda}_s|\rangle=\sum_{p=1}^{P}\psi((\nu_s-p+1)/2)+P\ln 2+\ln|\boldsymbol{B}_s|,$$
$$\langle(\boldsymbol{x}_{ji}-\boldsymbol{\mu}_s)^T\boldsymbol{\Lambda}_s(\boldsymbol{x}_{ji}-\boldsymbol{\mu}_s)\rangle=(\hat{\boldsymbol{x}}_{ji}-\boldsymbol{m}_s)^T\nu_s\boldsymbol{B}_s(\hat{\boldsymbol{x}}_{ji}-\boldsymbol{m}_s)+P/u_s+\mathrm{tr}(\nu_s\boldsymbol{B}_s\hat{\boldsymbol{\Omega}}_{ji}).$$

10. $q(\boldsymbol{w}_s|\boldsymbol{\mu}_s^w,\boldsymbol{\Sigma}_s^w)$

$$\begin{aligned}\boldsymbol{\Sigma}_s^w \quad &= \quad \left(\sum_{j=1}^{J}\sum_{i=1}^{n_j}E\sigma_{jis}(\hat{\boldsymbol{x}^b}_{ji}\hat{\boldsymbol{x}^b}^T_{ji}+\hat{\boldsymbol{\Omega}}_{ji})+\mathrm{diag}(\langle\boldsymbol{\lambda}\rangle)\right)^{-1}, \\ \boldsymbol{\mu}_s^w \quad &= \quad \boldsymbol{\Sigma}_s^w\left(\sum_{j=1}^{J}\sum_{i=1}^{n_j}E\sigma_{jis}\hat{\boldsymbol{x}^b}_{ji}\langle t_{ji}\rangle+\mathrm{diag}(\langle\boldsymbol{\lambda}\rangle)\phi\right).\end{aligned}$$

$$\langle\boldsymbol{w}_s\rangle=\boldsymbol{\mu}_s^w,\qquad \langle\boldsymbol{w}_s\boldsymbol{w}_s^T\rangle=\boldsymbol{\Sigma}_s^w+\boldsymbol{\mu}_s^w(\boldsymbol{\mu}_s^w)^T.$$

11. $q(\lambda_p|a_p,b_p)$, $\langle\lambda_p\rangle = a_p/b_p$.

$$\phi_p = \sum_{s=1}^{S}\langle w_{sp}\rangle/\gamma, \quad \gamma = \gamma_0 + S,$$
$$a_p = a_0 + \frac{S}{2}, \quad b_p = b_0 + \frac{1}{2}\sum_{s=1}^{S}\langle W_{sp}^2\rangle - \frac{1}{2}\gamma\phi_p^2.$$

References

J. H. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88:669–679, 1993.

R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

H. Attias. A variational Bayesian framework for graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD dissertation, University College London, Gatsby Computational Neuroscience Unit, 2003.

D. M. Blei and M. I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–144, 2006.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller. Max-margin classification of data with absent features. *Journal of Machine Learning Research*, 9:1–21, 2008.

L. M. Collins, J. L. Schafer, and C. M. Kam. A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods*, 6(4):330–351, 2001.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:1–38, 1977.

U. Dick, P. Haider, and T. Scheffer. Learning from incomplete data with infinite imputations. In *International Conference on Machine Learning (ICML)*, 2008.

D. B. Dunson, N. Pillai, and J.-H. Park. Bayesian density regression. *Journal of the Royal Statistical Society: Series B*, 69, 2007.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

T. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1: 209–230, 1973.

A. E. Gelfand, S. E. Hills, A. Racine-Poon, and A. F. M. Smith. Illustration of Bayesian inference in normal data models using Gibbs sampling. *Journal of American Statistical Association*, 85: 972–985, 1990.

Z. Ghahramani and M. J. Beal. Variational inference for Bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems (NIPS) 12*, pages 449–455. MIT Press, 2000.

Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto, 1996.

Z. Ghahramani and M. I. Jordan. Learning from incomplete data. Technical report, Massachusetts Institute of Technology, 1994.

T. Graepel. Kernel matrix completion by semidefinite programming. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 694–699, 2002.

J. Hanley and B. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

L. Hannah, D. Blei, and W. Powell. Dirichlet process mixtures of generalized linear models. In *Artificial Intelligence and Statistics (AISTATS)*, pages 313–320, 2010.

J. Ibrahim. Incomplete data in generalized linear models. *Journal of the American Statistical Association*, 85:765–769, 1990.

H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96:161–173, 2001.

R. A. Jacobs, M. I. Jordon, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

K. Kurihara, M. Welling, and Y. W. Teh. Collapsed variational Dirichlet process mixture models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2796–2801, 2007.

Percy Liang and Michael I. Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 584–591, 2008.

X. Liao, H. Li, and L. Carin. Quadratically gated mixture of experts for incomplete data classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 553–560, 2007.

Q. Liu, X. Liao, and L. Carin. Semi-supervised multitask learning. In *Neural Information Processing Systems*, 2007.

S. N. MacEachern and P. Müller. Estimating mixture of Dirichlet process models. *Journal of Computational and Graphical Statistics*, 7, 1998.

E. Meeds and S. Osindero. An alternative infinite mixture of Gaussian process experts. In *NIPS 18*, pages 883–890. MIT Press, 2006.

P. Müller, A. Erkanli, and M. West. Bayesian curve fitting using multivariate normal mixtures. *Biometrika*, 83:67–79, 1996.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical report, Department of Computer Science, University of Toronto, 1993.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases. *http://www.ics.uci.edu/∼mlearn/MLRepository.html*, 1998.

A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *NIPS 14*. MIT Press, 2002.

A. Rodríguez, D. B. Dunson, and A. E. Gelfang. Bayesian nonparametric functional data analysis through density estimation. *Biometrika*, 96, 2009.

D. B. Rubin. Inference and missing data. *Biometrika*, 63:581–592, 1976.

D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons, Inc., 1987.

J. L. Schafer and J. W. Graham. Missing data: Our view of the state of the art. *Psychological Methods*, 7:147–177, 2002.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 1:639–650, 1994.

B. Shahbaba and R. Neal. Nonlinear models using Dirichlet process mixtures. *Journal of Machine Learning Research*, 10:1829–1850, 2009.

P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7:1283–1314, 2006.

A. Smola, S. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

Y. W. Teh, M. J. Beal M. I. Jordan, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581, 2006.

M. E. Tipping. The relevance vector machine. In T. K. Leen S. A. Solla and K. R. Müller, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 12, pages 652–658. MIT Press, 2000.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

X. Wang, A. Li, Z. Jiang, and H. Feng. Missing value estimation for DNA microarray gene expression data by support vector regression imputation and orthogonal coding scheme. *BMC Bioinformatics*, 7:32, 2006.

S. R. Waterhouse and A. J. Robinson. Classification using hierarchical mixtures of experts. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IV*, pages 177–186, 1994.

M. West, P. Müller, and M. D. Escobar. Hierarchical priors and mixture models, with application in regression and density estimation. In P. R. Freeman and A. F. Smith, editors, *Aspects of Uncertainty*, pages 363–386. John Wiley, 1994.

D. Williams and L. Carin. Analytical kernel matrix completion with incomplete multi-view data. In *Proceedings of the International Conference on Machine Learning (ICML) Workshop on Learning with Multiple Views*, pages 80–86, 2005.

D. Williams, X. Liao, Y. Xue, L. Carin, and B. Krishnapuram. On classification with incomplete data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):427–436, 2007.

L. Xu, M. I. Jordan, and G. E. Hinton. An alternative model for mixtures of experts. In *Advances in Neural Information Processing Systems (NIPS) 7*, pages 633–640, 1995.

Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

K. Yu, A. Schwaighofer, V. Tresp, W.-Y. Ma, and H. Zhang. Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical Bayes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 616–623, 2003.

J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *Advances in Neural Information Processing Systems*, 2006.

Y. Zhang, L. M. Collins, H. Yu, C. Baum, and L. Carin. Sensing of unexploded ordnance with magnetometer and induction data: theory and signal processing. *IEEE Transactions on Geoscience and Remote Sensing*, 41(5):1005–1015, 2003.

X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Journal of Machine Learning Research 11 (2010) 3313-3332 Submitted 1/10; Revised 6/10; Published 11/10

Maximum Likelihood in Cost-Sensitive Learning: Model Specification, Approximations, and Upper Bounds

Jacek P. Dmochowski JDMOCHOWSKI@CCNY.CUNY.EDU
Department of Biomedical Engineering
City College of New York – City University of New York
New York, NY 10031, USA

Paul Sajda PSAJDA@COLUMBIA.EDU
Department of Biomedical Engineering
Columbia University
New York, NY 10027, USA

Lucas C. Parra PARRA@CCNY.CUNY.EDU
Department of Biomedical Engineering
City College of New York – City University of New York
New York, NY 10031, USA

Editor: Charles Elkan

Abstract

The presence of asymmetry in the misclassification costs or class prevalences is a common occurrence in the pattern classification domain. While much interest has been devoted to the study of *cost-sensitive learning* techniques, the relationship between cost-sensitive learning and the specification of the model set in a parametric estimation framework remains somewhat unclear. To that end, we differentiate between the case of the model including the true posterior, and that in which the model is misspecified. In the former case, it is shown that thresholding the maximum likelihood (ML) estimate is an asymptotically optimal solution to the risk minimization problem. On the other hand, under model misspecification, it is demonstrated that thresholded ML is suboptimal and that the risk-minimizing solution varies with the misclassification cost ratio. Moreover, we analytically show that the negative weighted log likelihood (Elkan, 2001) is a tight, convex upper bound of the empirical loss. Coupled with empirical results on several real-world data sets, we argue that weighted ML is the preferred cost-sensitive technique.

Keywords: empirical risk minimization, loss function, cost-sensitive learning, imbalanced data sets

1. Introduction

Pattern classifiers make decisions; when those decisions are wrong, a loss is incurred. Thus, the ultimate goal of a classifier is to minimize the loss. When put into probabilistic terms, the mathematical expectation of the loss is called the *risk*, and is related to the classifier's error rates. In the case of a binary classification this can be written as (Duda et al., 2001):

$$\text{risk} = p(+1)\,c(+1)\,p(\text{error}|+1) + p(-1)\,c(-1)\,p(\text{error}|-1)\,, \tag{1}$$

where $c(+1)$ and $c(-1)$ denote the costs of a false negative and false positive, respectively, $p(+1)$ and $p(-1)$ are the prior probabilities for classes $y=+1$ and $y=-1$, $p(\text{error}|+1)$ is the false

negative rate, and $p(\text{error}|-1)$ is the false positive rate. Notice that the false positive and negative rates are the only terms which depend on the classifier parameters, whereas the misclassification costs and class priors are typically constants of the classification problem (later, we consider the case of example-dependent costs). The class priors are coupled with the costs of misclassification in the expression for expected loss. Thus, the risk minimization problem is uniquely defined by the ratio $\frac{p(+1)c(+1)}{p(-1)c(-1)}$; that is, even though the priors and costs may vary, as long as this ratio stays constant, the optimization problem is unchanged.

The term *cost-sensitive learning* (Elkan, 2001) has been attached to classification environments in which $c(+1) \neq c(-1)$. On the other hand, *classification with imbalanced data sets* (Chawla and Japkowicz, 2004) refers to the case where $p(+1) \neq p(-1)$. The presence of at least one of these asymmetries has been referred to by some as the "nonstandard" case (Lin et al., 2002), even though the situation is rather common in practice. In any case, these two problems may be unified simply by stating that the goal of the classification is to minimize the risk, as opposed to the conventional error rate:

$$\text{error rate} = p(+1)\, p(\text{error}|+1) + p(-1)\, p(\text{error}|-1).$$

A classifier that is designed to minimize the error rate will generally yield a high expected loss when applied to the case $c(+1) \neq c(-1)$, as the error-minimizing classifier will under-emphasize the more costly class. The problem may be exacerbated if the class prevalences are also skewed, and in the extreme case, the algorithm yields a trivial classifier which always selects the common class.

Minimizing risk is synonymous with optimally trading off the false negative and false positive rates. The trade-off between the false positive rate and false negative rate is precisely depicted by receiver operating characteristic (ROC) curves (Provost and Fawcett, 1997; Fawcett, 2004; Egan, 1975). Thus, ROC curves are well-suited to evaluating the expected loss of a classifier across the range of misclassification costs. However, "reading off" the expected loss from an ROC graph is not straightforward, and Drummond and Holte (2000) proposed cost curves as an explicit visualization of a classifier's risk for varying misclassification costs and class priors. Since the ratio $\frac{p(+1)c(+1)}{p(-1)c(-1)}$ is unbounded, the curves instead show the risk as a function of the *probability cost function* (pcf):

$$\text{pcf} = \frac{p(+1)c(+1)}{p(+1)c(+1) + p(-1)c(-1)}.$$

Cost curves facilitate the quantification of the reduction in loss offered by a cost-sensitive learning algorithm.

Several methodologies have been developed in the effort to design risk-minimizing classifiers. The simplest approach is to modify the threshold of an existing, cost-insensitive classifier. If the classifier is based on the log of the ratio of true class posterior probabilities, the threshold should be modified by a value equal to the log of the ratio of misclassification costs (Duda et al., 2001). In practice, the true class-conditional probabilities are unknown. Nevertheless, shifting the threshold by the corresponding amount has become a common heuristic (Elkan, 2001; Lin et al., 2002). Elkan (2001) proposes handling asymmetric misclassification costs by retraining the classifier on a training set in which the proportion of positive and negative examples is matched to the ratio of misclassification costs. Alternatively, if an algorithm may apply weights to the training examples, the negative examples should be weighted by a value corresponding to the asymmetry in misclassification costs. Maloof (2003) points out that although the problems of imbalanced data sets and

varying misclassification costs are separate problems, they may be tackled in very similar ways. He shows empirically that oversampling the less prevalent class (or undersampling the more prevalent class) is a procedure which yields results virtually identical to adjusting the decision threshold.

Domingos (1999) proposes a technique to relabel the training data in such a way that the relabeled data set may be trained using a standard (cost-insensitive) technique to yield a cost-sensitive classifier. The posteriors for the *labeled* examples are estimated via bagging and then used in conjunction with the Bayesian minimum risk criterion to assign new labels to the supervised data. Margineantu (2000) analyzes the approach of Domingos (1999) and suggests ways of improving the class probability estimates of the training data. Dudik and Phillips (2009) address the class imbalance problem by proposing a method which attempts to minimize loss assuming the worst-case class proportions. Masnadi-Shirazi and Vasconcelos (2010) describe a cost-sensitive version of the popular support vector machine.

Some work has been devoted to the case of example-dependent costs (Zadrozny and Elkan, 2001; Zadrozny et al., 2003). Moreover, some authors have advocated for maximizing benefits rather than minimizing costs (Elkan, 2001).

In Guerrero-Curieses et al. (2004), the authors examine loss functions which are minimized by the true class posterior probabilities; moreover, it is pointed out that the corresponding optimization algorithms should focus on training points near the decision boundary.

It is also important to point out that risk minimization is a diverse problem spanning multiple research communities; in particular, significant contributions to the problem have been made in the econometrics literature. To that end, Elliott and Lieli (2007) examine a problem analogous to cost-sensitive learning, namely the determination of a profit maximizing decision scheme by a lender. It is noted therein that to construct the minimum risk decision, the model density need not match the true density; rather, it is only required that the classifier output from the model density falls on the same side of the threshold as the classifier output using the true density. Moreover, the authors use a new loss function, namely an affine transformation of the expected utility (risk), and show an empirical advantage over traditional methods.

While a plethora of cost-sensitive methods has been investigated, it remains unclear under what conditions shifting the threshold of an existing cost-insensitive classifier is an appropriate solution. The distinction between the case of the model family including the true posterior, versus that of "misspecification" (the model does not contain the truth), has large implications on the resulting cost-sensitive learning process.

In the former case, shifting the threshold of the maximum likelihood (ML) solution is an asymptotically optimal solution to the risk minimization problem, and in the following we provide a proof of this important point. This means that when employing an expressive family which contains the true posterior, the cost-sensitive learning problem becomes one of density estimation, and the costs affect only the threshold, not the estimator. This may lead one to use a rich model set leading to complex classifiers. However, the choice to employ a simple classifier brings many advantages: ease of implementation, a lesser number of parameters to estimate, a reduced risk of over-fitting, and consequently simplified regularization procedures. Coupled with the complexity of real-world data sets, misspecified models are frequently encountered in practice. In this case, we demonstrate that thresholded ML is suboptimal, and that the minimum risk solution varies with the ratio of misclassification costs.

The problems with minimizing the true empirical risk, a non-smooth function, are well-known: for zero-one loss, the idea of smoothing out the indicator function appears in Horowitz (1992). In

this paper, we employ a sigmoidal approximation of the empirical risk to yield a novel minimizer of the loss under asymmetric misclassification cost values. Rather than argue for its optimality, this estimator is used as a basis for comparison and to argue for the relative merits of existing cost-sensitive techniques. We show analytically that the negative weighted log likelihood serves as an upper bound to the sigmoidal empirical risk. Based on the convexity of the negative weighted log likelihood and forthcoming numerical results, we will argue that weighted ML is generally the preferred technique.

2. Classification Model

In the following, we adopt a probabilistic model for the classification task: assume that the true posterior probability $p(y|\mathbf{x})$ of class $y \in \{-1,+1\}$ given received feature vector $\mathbf{x} \in \mathbb{R}^D$ is known. Let $c(y,\mathbf{x})$ denote the cost of a misclassification when the true class is y for feature $\mathbf{x}$, minus the cost of a correct prediction. Note that in general, c is feature-dependent, although in many applications, $c(y,\mathbf{x}) = c(y)$. If there is also no cost for a correct decision, then $c(y)$ is simply the cost of a false positive ($y=-1$) or false negative ($y=+1$). The optimal Bayesian decision rule is to predict $\hat{y}=+1$ if (Duda et al., 2001):

$$\frac{p(+1|\mathbf{x})}{p(-1|\mathbf{x})} > c_o(\mathbf{x}),$$

where $c_o(\mathbf{x}) = \frac{c(-1,\mathbf{x})}{c(+1,\mathbf{x})} > 0$. The optimal decision rule may be written as:

$$\hat{y}(\mathbf{x}) = \operatorname{sgn}\left[f(\mathbf{x}) - \ln c_o(\mathbf{x})\right], \tag{2}$$

where $\hat{y}(\mathbf{x})$ is the predicted class given feature vector $\mathbf{x}$, sgn is the signum function $\operatorname{sgn}(x) = \begin{cases} 1 & x>0 \\ -1 & x\leq 0 \end{cases}$, and $f(\mathbf{x})$ is the discriminant function:

$$f(\mathbf{x}) = \ln \frac{p(+1|\mathbf{x})}{p(-1|\mathbf{x})}.$$

It should be noted that the argument of the signum function may be written in log units due to the nonnegativity of the ratio of posteriors and the optimal threshold $c_o(\mathbf{x})$.

In practice, we do not have access to the true class posteriors, but rather estimate their values from available training data. The estimate is denoted by $p(y|\mathbf{x};\theta)$, where $\theta \in \Theta$ is a vector parameterizing the *model* posterior, and Θ is termed the model set. If the true posterior is in the model set, denote the true value of θ by θ^*, such that $p(y|\mathbf{x}) = p(y|\mathbf{x};\theta^*)$. The model discriminant is written as $f(\mathbf{x},\theta) = \ln \frac{p(+1|\mathbf{x};\theta)}{p(-1|\mathbf{x};\theta)}$, and the classifier takes the form:

$$\hat{y}(\mathbf{x},\theta) = \operatorname{sgn}\left[f(\mathbf{x},\theta) - \ln c_o(\mathbf{x})\right]. \tag{3}$$

This paper is concerned with methods of estimating θ to minimize risk, and their relation to the specification of the model set. In order to treat these estimation methods, we briefly outline the risk minimization framework which allows for the subsequent analysis of the various cost-sensitive loss functions.

3. Risk Minimization for Cost-Sensitive Learning

Risk minimization is concerned with choosing a function from a set $\{\hat{y}(\mathbf{x},\theta),\theta\in\Theta\}$ to minimize the resulting *risk functional*

$$R(\theta)=\int\int L(y,\mathbf{x},\theta)\,p(\mathbf{x},y)\,d\mathbf{x}dy,$$

where $L(y,\mathbf{x},\theta)$ quantifies the loss incurred by the classifier $\hat{y}(\mathbf{x},\theta)$ in response to labeled data $(\mathbf{x},y)$. Note that the loss L varies with the feature $\mathbf{x}$. To ease notation throughout the rest of the paper, the dependence of $\hat{y}$ on $\mathbf{x}$ and θ is implied.

The problems of regression, density estimation, and pattern recognition may all be formulated within the context of risk minimization, simply by altering the loss function L, as outlined in Vapnik (1998, 1999). In the case of error-minimizing pattern recognition, the classical zero-one loss function is given by:

$$L(y,\mathbf{x},\theta)=\mathbb{1}(y\neq\hat{y}),$$

where $\mathbb{1}(\Phi)$ is the indicator function which equals one when Φ is true and zero otherwise.

Since we do not have access to the true density $p(\mathbf{x},y)$, the empirical risk minimization (ERM) approach substitutes the empirical density:

$$p_{\text{emp}}(\mathbf{x},y)=\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}(\mathbf{x}=\mathbf{x}_n)\,\mathbb{1}(y=y_n),$$

where $\mathcal{D}=(\mathbf{x}_n,y_n)_{n=1}^{N}$ is a set of N labeled observations which are independent and identically distributed samples drawn from the true joint density $p(\mathbf{x},y)$, leading to the following expression for the *empirical risk*:

$$R_{\text{emp}}(\theta)=\frac{1}{N}\sum_{n=1}^{N}L(y_n,\mathbf{x}_n,\theta). \tag{4}$$

In order to design a cost-sensitive classifier, a loss function modeling the asymmetry in misclassification costs is required. Several alternatives exist. In the following subsections, we describe these loss functions.

3.1 Thresholded Maximum Likelihood

The traditional (cost-insensitive) ML loss function is given by Vapnik (1998):

$$L(y,\mathbf{x},\theta)=-\ln p(y|\mathbf{x};\theta),$$

leading to the following expression for the empirical risk:

$$R^{\text{ml}}_{\text{emp}}(\theta)=-\frac{1}{N}\sum_{n}\ln p(y_n|\mathbf{x}_n;\theta). \tag{5}$$

The minimizer of (5) is the well-known ML estimate (Duda et al., 2001):

$$\begin{aligned}\hat{\theta}_{\mathrm{ML}} &= \arg\max_{\theta\in\Theta} \ln \prod_{n=1}^{N} p\left(y_n|\mathbf{x}_n;\theta\right) \\ &= \arg\max_{\theta\in\Theta} \ln \prod_{n=1}^{N} \frac{1}{1+\frac{p\left(-y_n|\mathbf{x}_n;\theta\right)}{p\left(y_n|\mathbf{x}_n;\theta\right)}} \\ &= \arg\max_{\theta\in\Theta} \sum_{n=1}^{N} \ln\left[\frac{1}{1+e^{-y_n f\left(\mathbf{x}_n,\theta\right)}}\right],\end{aligned}$$

where the second step follows from Bayes' rule. If the model set Θ contains the true parameter θ^*, it follows that in the asymptotic limit, we have $\lim_{N\to\infty}\hat{\theta}_{\mathrm{ML}} = \theta^*$ (Kay, 1993). From (2) and (3), if we have knowledge of θ^*, then a threshold shift of $\ln c_o(\mathbf{x})$ yields the optimal classifier. Assuming continuity of the log likelihood (in θ), we have that $\lim_{N\to\infty}\hat{y}\left(\mathbf{x},\hat{\theta}_{\mathrm{ML}}\right) = \hat{y}\left(\mathbf{x},\theta^*\right)$, and thus the thresholded ML estimate yields the minimum risk decision rule for all cost ratios. Once the ML estimate is available, the cost-sensitive classifier for any cost ratio may be formed by appropriately adjusting the threshold. There is no need to retrain the classifier if the cost ratio changes. In the case of a generalized linear model for $p(y|\mathbf{x},\theta)$, it may easily be shown (McCullagh and Nelder, 1989; Parra et al., 2005) that the risk function is convex, and an iteratively reweighted least squares (IRLS) algorithm locates the optimal linear classifier often within a few iterations.

Unfortunately, in many real-world classification problems, the model set (for example, the set of all hyperplanes) does not contain the true posterior. Notice, for example, that even in the simple case of Gaussian data, the linear discriminant is only optimal in the case of equal covariance matrices; nevertheless, linear classifiers are heavily used. (For a comprehensive treatment of misspecified models in ML, refer to White, 1982.) In such cases, the classifier in the model set which minimizes risk will vary with the pcf. As a result, a shift in threshold of the ML solution will yield a sub-optimal classifier.

3.2 Example: Minimum Risk Hyperplane for Gaussian Data

To illustrate this point, we consider the instructive case of Gaussian densities with unequal covariances and a linear classifier function. The purpose of this exercise is not to argue for a simple Gaussian model or a linear classifier but rather to demonstrate in an analytically tractable case the problem that arises with thresholded ML when the model is misspecified. It is assumed that $c\left(y,\mathbf{x}\right) = c(y)$.

Consider a linear classifier of the form $f\left(\mathbf{x};\theta\right) = \theta^T\mathbf{x} - b$, and assume Gaussian class-conditional densities:

$$p\left(\mathbf{x}|y\right) = \frac{1}{(2\pi)^{D/2}\left|\Sigma_y\right|^{1/2}} e^{-\frac{1}{2}\left(\mathbf{x}-\mu_y\right)^T\Sigma_y^{-1}\left(\mathbf{x}-\mu_y\right)},\ y\in\{-1,+1\}.$$

Note that by the normality of $\mathbf{x}|y$, $\theta^T\mathbf{x}\sim\mathcal{N}\left(\theta^T\mu_y,\theta^T\Sigma_y\theta\right)$. Thus, we have:

$$\begin{aligned}p\left(\text{error}|y\right) &= p\left[y\left(\theta^T\mathbf{x}-b\right)<0\right] \\ &= \frac{1}{2}\left[1+y\cdot\operatorname{erf}\left(\frac{b-\theta^T\mu_y}{\sqrt{2\theta^T\Sigma_y\theta}}\right)\right].\end{aligned} \tag{6}$$

Substituting (6) into (1), the expression for the expected loss takes the form:

$$R(\theta,b) = \frac{c(+1)p(+1)}{2}\left[1+\text{erf}\left(\frac{b-\theta^T\mu_+}{\sqrt{2\theta^T\Sigma_+\theta}}\right)\right] + \frac{c(-1)p(-1)}{2}\text{erfc}\left(\frac{b-\theta^T\mu_-}{\sqrt{2\theta^T\Sigma_-\theta}}\right). \quad (7)$$

The optimal hyperplane is the one which minimizes the risk: $\arg\min_{\theta,b} R(\theta,b)$.

Below, we illustrate an example where the parameters of the data are given by:

$$\mu_+ = \begin{bmatrix} 0.5 & 0 \end{bmatrix}^T,\ \mu_- = \begin{bmatrix} 0 & 0.5 \end{bmatrix}^T,\ \Sigma_+ = \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix},\ \Sigma_- = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix}.$$

In this case of unequal covariance, the optimal ML classification function is not linear but instead a quadric (Duda et al., 2001, Chapter 2).

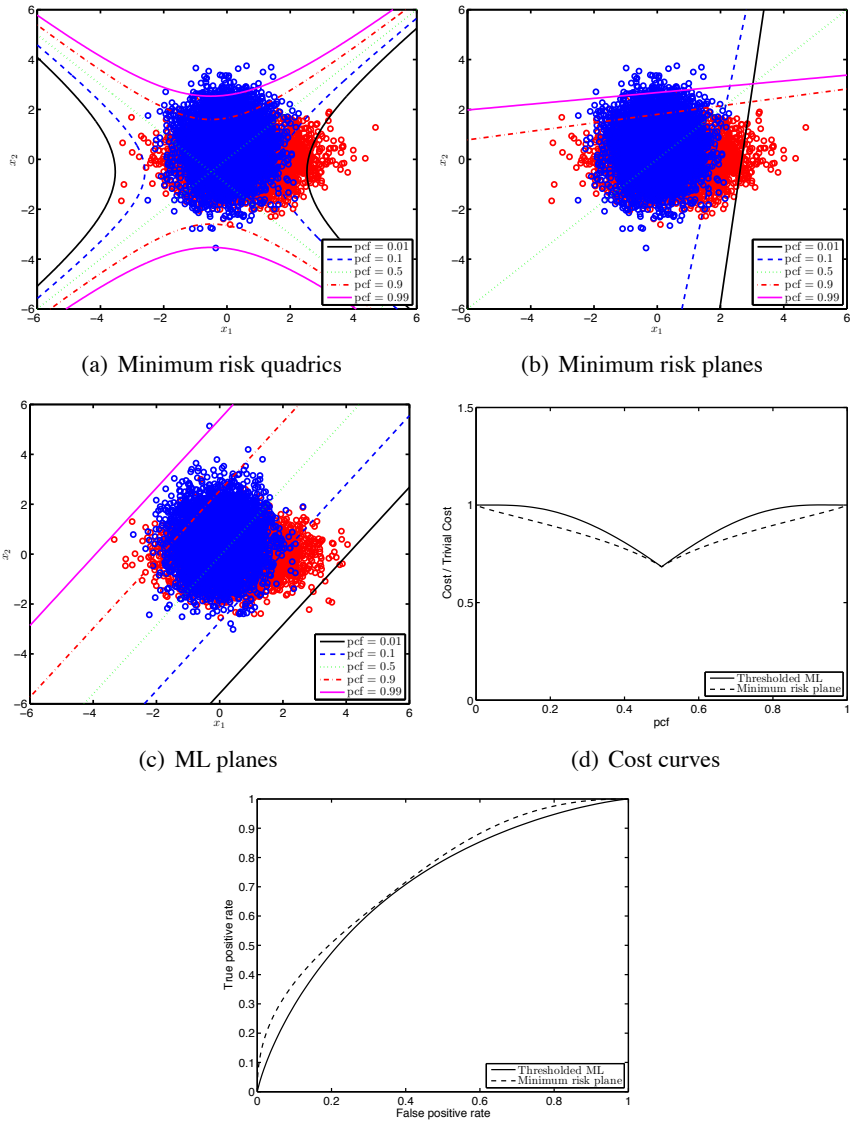
Figure 1(a) displays the minimum risk quadrics for various values of the pcf, where it is assumed that $p(+1) = p(-1)$. The quadrics are related to each other via a threshold shift. On the other hand, Figure 1(b) depicts the minimum risk planes for the same data, which were computed by minimizing (7) using numerical optimization techniques. It is clear that the direction of the optimal plane is a function of the pcf, and a threshold shift of the minimum error plane is not an optimal solution to the risk minimization problem. Figure 1 (c) displays the threshold-shifted ML solutions for various values of the pcf. The suboptimality of the ML approach is readily apparent by contrasting Figure 1(b) with Figure 1(c). Notice that at the extremes of the pcf, the optimal planes are orthogonal to each other. Meanwhile, the ML plane has unit slope for all pcf. The risks obtained by applying the ML and minimum risk planes to the given data are shown in Figure 1(d). In the figure, we normalize the raw risk of (1) by the "trivial risk", which is defined as the risk achieved by the scheme:

$$\hat{y}_{\text{trivial}} = \text{sgn}\left[p(+1)c(+1) - p(-1)c(-1)\right].$$

We call this the "trivial risk" because the decision rule is feature-independent and is strictly a function of the class priors and misclassification costs. A normalized risk less than 1 indicates that the classification scheme yields a "savings" over the a priori decision rule. The normalization allows us to quantify the "percentage of savings" achieved by employing a "smart" decision rule.

The curves were generated by averaging over 1000 ensembles, where each ensemble consisted of $N = 1000$ training samples. The ML classifier was trained on each ensemble and the resulting risk computed by substituting the solution into (7). The risk margin between the threshold-shifted ML solution and that of the minimum risk plane is what is "available" for cost-sensitive learning algorithms to improve upon. These methods attempt to learn, for each pcf, the minimum risk plane shown in Figure 1(b), to achieve the dashed cost curve in Figure 1(d).

The difference between the threshold-shifted ML and cost-sensitive paradigms may be understood in terms of ROC analysis—Figure 1 (e) depicts the ROC curves for the thresholded ML and minimum risk classifiers. In the ML method, the ROC curve is generated by sweeping the threshold of the base classifier across the real line and computing the corresponding error rates. In the cost-sensitive paradigm, each point on the ROC curve corresponds to a distinct classifier which is computed by minimizing (7) for a specific ratio of misclassification costs, resulting in values for the true and false positive rates. Note that one may also produce a *family* of ROC curves by sweeping the threshold of each of these distinct cost-sensitive classifiers, although this is not shown in the figure.



(a) Minimum risk quadrics

(b) Minimum risk planes

(c) ML planes

(d) Cost curves

(e) ROC curves

Figure 1: Minimum risk classification of Gaussian data with unequal covariance matrices.

3.3 Relabeled Examples

One heuristic to cost-sensitive classification is to modify the training labels to achieve a balance in class prevalence. In terms of an ERM loss function, this may be written as:

$$L(y,\mathbf{x},\theta) = -\ln p\left[g(\mathbf{x})|\mathbf{x};\theta\right],$$

where the function $g(\mathbf{x}) : \mathbb{R}^D \rightarrow \{-1,+1\}$, produces a new label for each training sample according to some criterion. Domingos (1999) proposes MetaCost, which reassigns labels according to the

Bayesian minimum risk criterion:

$$g(\mathbf{x}) = \arg \max_{y \in \{-1,+1\}} c(y) \hat{p}(y|\mathbf{x}), \tag{8}$$

where $\hat{p}(y|\mathbf{x})$ is an estimate of the class posterior probability which is obtained using bagging (Breiman, 1996). It is typically the examples near the boundary which are re-labeled. The empirical risk follows as:

$$R^{\text{rel}}_{\text{emp}}(\theta) = -\frac{1}{N} \sum_n \ln p\left[g(\mathbf{x}_n)|\mathbf{x}_n;\theta\right].$$

As the re-labeling of (8) varies with the ratio of misclassification costs, the resulting cost-sensitive classifier is a function of the pcf and thus has the ability to yield the minimum risk estimator. The success of the method hinges on the estimation of the posterior probabilities; in the best case scenario, the re-labeling results in the cost-sensitive boundary approaching the minimum risk boundary. In contrast to the forthcoming methods, MetaCost does not reweight examples, and thus the risk function will not dominated by the examples of a rare but costly class. The technique may be used as a cost-sensitive pre-processing to any classification technique, and not just ML estimation of the posterior. In the case of ML, we maximize the log-likelihood but with the altered labels.

3.4 Weighted Likelihood

A standard approach for developing cost-sensitive classifiers is to weight the training examples according to the "costliness" of misclassifying that example. This procedure may be viewed in terms of an ERM loss function (Elkan, 2001; Zadrozny et al., 2003):

$$L(y,\mathbf{x},\theta) = -c(y,\mathbf{x}) \ln p(y|\mathbf{x};\theta),$$

such that the corresponding empirical risk takes the form:

$$R^{\text{wml}}_{\text{emp}}(\theta) = -\frac{1}{N} \sum_n c(y_n,\mathbf{x}_n) \ln p(y_n|\mathbf{x}_n;\theta). \tag{9}$$

Weighting the log likelihood has previously been studied as a tool to handle misspecification (Shimodaira, 2000). Note that such weighting of examples is equivalent to modifying the proportion of examples in the training set according to the weightings $c(y,\mathbf{x})$. If these weightings change, so does the cost function, and thus the classifier needs to be retrained. In principle, this technique allows the classification to choose the model in Θ which minimizes risk for the specified cost matrix. Moreover, example-weighting may easily be incorporated into the IRLS algorithm, yielding an iterative reweighted *weighted* least-squares scheme (McCullagh and Nelder, 1989) which minimizes (9)—in the appendix, we provide a MATLAB implementation.

Notice that if the misclassification costs are highly asymmetric, the more "costly" examples will be heavily emphasized in the empirical risk function. Furthermore, if there are only a few such examples, the classifier is at an increased risk of overfitting, since the *effective* number of examples is much less than N. This issue plagues any cost-sensitive method which weights the examples based on cost.

3.5 Sigmoidal Empirical Risk

In order to relate the risk of (1) with the empirical risk (4), the appropriate loss function is found to be:

$$L(y,\mathbf{x},\theta) = c(y,\mathbf{x})\mathbb{1}(\hat{y} \neq y).$$

Strict equivalence is achieved in the simplifying case of $c(y,\mathbf{x}) = c(y)\mathbb{1}(\hat{y} \neq y)$ assuming that there is no cost for a correct decision. Generally, the empirical risk follows as:

$$\begin{aligned} R_{\text{emp}}(\theta) &= \text{const.} + \frac{1}{N}\sum_n c(y_n,\mathbf{x}_n)\,\mathbb{1}(\hat{y}_n \neq y_n) \\ &= \text{const.} + \frac{1}{N}\sum_n c(y_n,\mathbf{x}_n)\,u[-y_n f(\mathbf{x}_n,\theta)] \end{aligned} \tag{10}$$

where the constant is a summation across the costs of a correct decision and $u(x)$ is the step function: $u(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$. Since our goal is to minimize (1), the optimization of the empirical risk under the direct loss of (10) is of great importance.

Elliott and Lieli (2007) propose to optimize a function closely related to (10) in an econometric context; employing the notation of this paper, the objective function maximized by Elliott and Lieli (2007) is written as:

$$R_{\text{el}}(\theta) = \sum_{n=1}^{N} y_n c(y_n,\mathbf{x}_n)\,\text{sgn}[f(\mathbf{x}_n,\theta) - \ln c_o(\mathbf{x}_n)], \tag{11}$$

and the authors propose simulated annealing to perform the optimization.

Note that both objective functions (10) and (11) are not differentiable due to the non-smoothness of u and sgn at zero, respectively. In the case of (10), we may approximate the step function with a sigmoid :

$$u(x) \approx \frac{1}{1+e^{-x}}. \tag{12}$$

Substituting (12) into (10), we obtain the following expression for the approximate empirical risk:

$$\tilde{R}_{\text{emp}}(\theta) = \frac{1}{N}\sum_n c(y_n,\mathbf{x}_n)\frac{1}{1+e^{y_n f(\mathbf{x}_n,\theta)}}.$$

The classifier θ which minimizes the empirical risk follows as:

$$\hat{\theta} = \arg\min_{\theta} \tilde{R}_{\text{emp}}(\theta). \tag{13}$$

The advantage of this approach is that it closely approximates (up to the ability of the sigmoid to approximate a step) the true empirical risk. On the other hand, the risk function is non-convex, complicating the minimization of (13).

3.6 Relating Sigmoidal Risk to Weighted ML

The need to minimize non-convex functions arises often in practice. A standard trick in optimizing a non-convex function is to optimize a convex upper bound of the original function. In this subsection, we show that the negative weighted log likelihood, a convex function, provides a tight upper bound of the sigmoidal empirical risk.

To see this, note the inequality:

$$z \leq -\ln(1-z), \;\; z \leq 1. \tag{14}$$

Substituting $z = \frac{1}{1+e^{y_n f(\mathbf{x}_n,\theta)}}$ into (14) results in:

$$\frac{1}{1+e^{y_n f(\mathbf{x}_n,\theta)}} \leq -\ln \frac{1}{1+e^{-y_n f(\mathbf{x}_n,\theta)}}.$$

Combining these inequalities over the training examples and assuming strict positivity of the weights $c(y_n, \mathbf{x}_n)$, we obtain:

$$\sum_n c(y_n,\mathbf{x}_n) \frac{1}{1+e^{y_n f(\mathbf{x}_n,\theta)}} \leq -\sum_n c(y_n,\mathbf{x}_n) \ln \frac{1}{1+e^{-y_n f(\mathbf{x}_n,\theta)}}.$$

As a result,

$$\tilde{R}_{\text{emp}}(\theta) \leq R^{\text{wml}}_{\text{emp}}(\theta).$$

This means that minimizing the weighted negative log-likelihood (9) minimizes an upper bound on the empirical risk. As will be shown numerically with upcoming examples, this bound is fairly tight (c.f., Figures 2 and 3). Since the negative weighted log-likelihood is convex, to circumvent the non-convexity of the sigmoidal empirical risk, one option is to employ the weighted likelihood loss function.

4. Experimental Evaluation

To assess the performance of the various cost-sensitive approaches (and its dependence on N), and to support the upper bound relationship of weighted ML to sigmoidal risk, we conducted an empirical evaluation of the various cost-sensitive learning approaches on several data sets. We first consider the case of example-independent and synthetic (i.e., exogenous to the features) costs. Later, we examine a data set where costs are endogenous and depend on the features. We employed a linear model set of the form $f(\mathbf{x},\theta) = \theta^T \mathbf{x} - b$. For all data sets, five-fold cross-validation was employed, and we plot the mean loss over the N examples (each example is used once for validation) along with standard errors of the mean.

The prevalence of the positive class is data-set dependent. The pcf was varied from 0.1 to 0.9 in increments of 0.1. The relationship between the misclassification cost ratio and the pcf is given by:

$$\frac{c(-1)}{c(+1)} = \frac{p(+1)}{p(-1)} \frac{(1-\text{pcf})}{\text{pcf}}.$$

Thus, a pcf of 0.5 corresponds to the case where the ratio of misclassification costs is inversely related to the ratio of class priors [i.e., $c(-1)p(-1) = c(+1)p(+1)$].

The generalization ability of all algorithms benefited from ℓ_2 regularization. Thus, the problem of cost-sensitive learning becomes one of penalized ERM:

$$\hat{\theta} = \arg\min_{\theta} \left\{ R_{\text{emp}}(\theta) + \frac{\lambda}{2} \|\theta\|^2 \right\}. \tag{15}$$

Where computationally feasible, the value of λ was determined using a nested cross-validation loop which tunes λ on the training set; the tuned value is then fed up to the outer cross-validation loop which evaluates performance on the test set.

The implementation of all cost-sensitive learning methods requires solving the optimization problem (15). For the 3 likelihood based methods, the Newton-Raphson IRLS algorithm (McCullagh and Nelder, 1989) was employed to solve the optimization. In order to solve the minimum risk optimization of (13), the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method (Fletcher, 2000) was employed in conjunction with multiple random restarts: a random starting vector is chosen, the BFGS algorithm is run, and the training risk evaluated. This process is repeated 100 times, and the final solution is selected as the classifier which yields the lowest training risk among the runs.

4.1 Gaussian Data

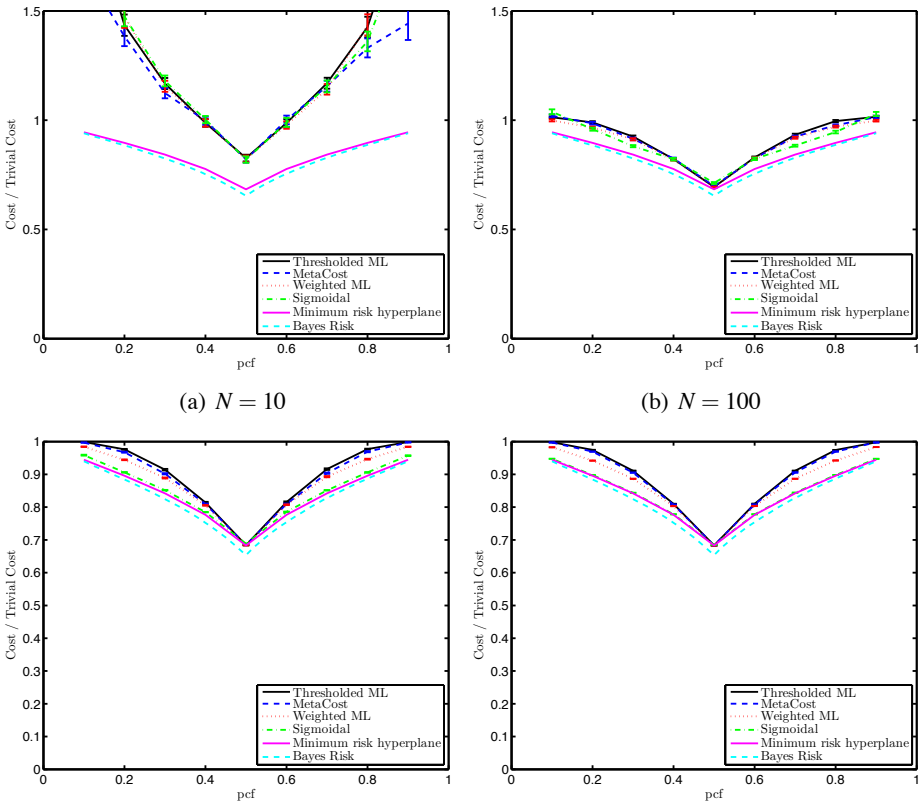
Before delving into real-world data sets, we evaluated the various cost-sensitive approaches on the Gaussian data described in Section 3. This example is instructive as we have a closed-form expression for the minimum attainable value of risk, and thus can evaluate the convergence properties with an increased sample size. Figure 2 depicts the cost curves[1] for various training sizes N. At $N = 10$, there is a substantial loss margin between the minimum risk plane and that which is achieved by the thresholded ML technique. However, it is clear that the cost-sensitive techniques are not able to provide reliable estimates of the minimum risk plane direction with such limited data. As the size of the training set increases, the sigmoidal risk estimator converges to the minimum risk plane. Notice, however, that with such large sample sizes, the thresholded ML technique is relatively adept at yielding risk values comparable to the true minimum. The reason for this is that the examples which are misclassified by thresholded ML and classified correctly by the the other techniques are mostly the low-cost examples (compare Fig. 1(b) with Fig. 1(c), for example). Also shown in all plots is the Bayes risk, which is the risk attained by the minimum risk quadric.

4.2 UCI Data

Next, we evaluate the classifiers on several real-world data sets obtained from the UCI database (Asuncion and Newman, 2007) as well as our previous work on the classification of electroencephalographic (EEG) data in a real-time detection task (Parra et al., 2008).[2] Table 1 summarizes the parameters used in the evaluation of these data sets.

From Figs. 3 (a) and (b), it is once again apparent that given a modest value of N, the benefits provided by cost-sensitive learners over the thresholded ML approach are not substantial. However, in Figs. 3 (c) and (d), one observes a tangible loss reduction of the sigmoidal risk estimator over

1. In addition to the ensemble-averaged risk, we also report standard errors of the mean, which follow as the sample standard deviation of the ensemble-averaged mean, divided by the square root of the number of ensembles.
2. Since only one "ensemble" is available in the experiments with real data, we treat the cost (at test time) of each example as an iid realization of the risk and report standard errors of the mean across examples.



(a) $N = 10$ (b) $N = 100$

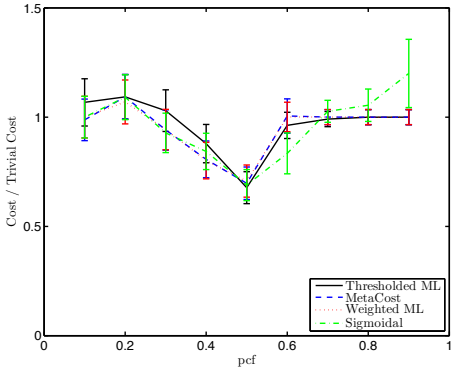
(c) $N = 1000$ (d) $N = 10000$

Figure 2: Cost curves for Gaussian data with varying training set sizes. In (d), the thresholded ML and MetaCost curves are nearly equivalent.

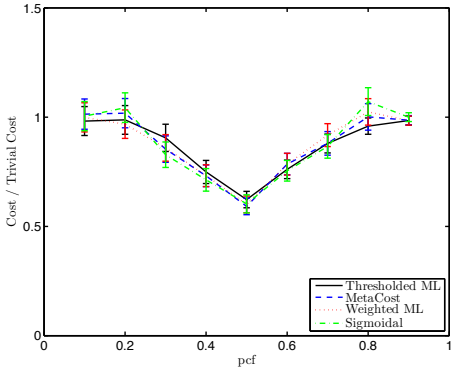
Data set	$N(+)$	$N(-)$	D	λ	$p(+1)$
Haberman	81	225	3	optimized	0.26
Transfusion	178	570	4	optimized	0.24
Magic	6688	12332	10	fixed (0.2)	0.35
Adult	7841	24720	14	fixed (0.2)	0.24
EEG	830	40608	15	fixed (2)	0.02

Table 1: Data set and regularization parameters. $N(+)$ and $N(-)$ refer to the number of positive and negative examples, respectively.

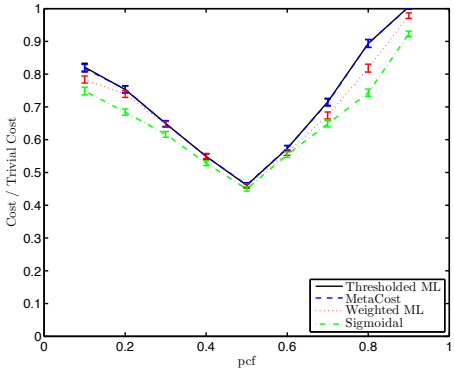
thresholded ML and MetaCost, whose curves overlap. Lastly, Fig. 3 (e) demonstrates the near-optimality of weighted ML and its close approximation of the sigmoidal risk minimizing solution. Note that for this heavily skewed data set, while the total number of examples is $N = 41438$, only 830 of these are positive exemplars. Note also that a skew in class prevalence leads to asymmetry in the resulting cost curves.



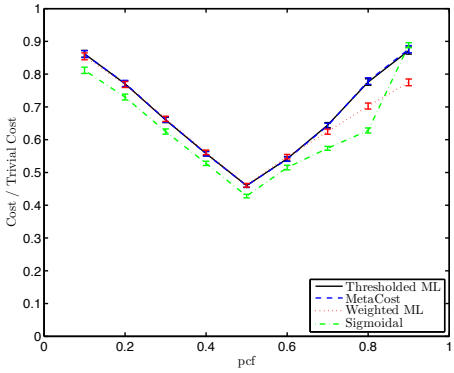
(a) Haberman



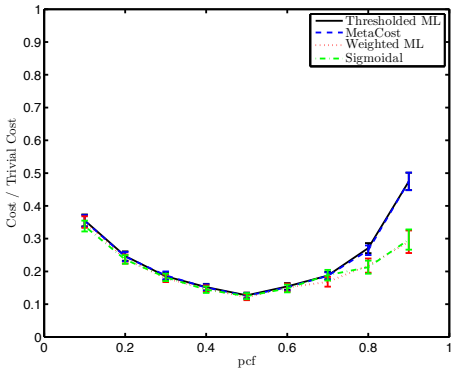
(b) Transfusion



(c) Magic



(d) Adult



(e) EEG

Figure 3: Cost curves (with standard errors of the means) for various real data sets with synthetic costs.

4.3 German Banking Data

Finally, we evaluated the risk minimizing classifiers on a publicly available data set collected by a German bank: http://www.stat.uni-muenchen.de/service/datenarchiv/kredit/kredit.html. The data set details the credit history and biographical information of $N = 1000$ past loan applicants, as well

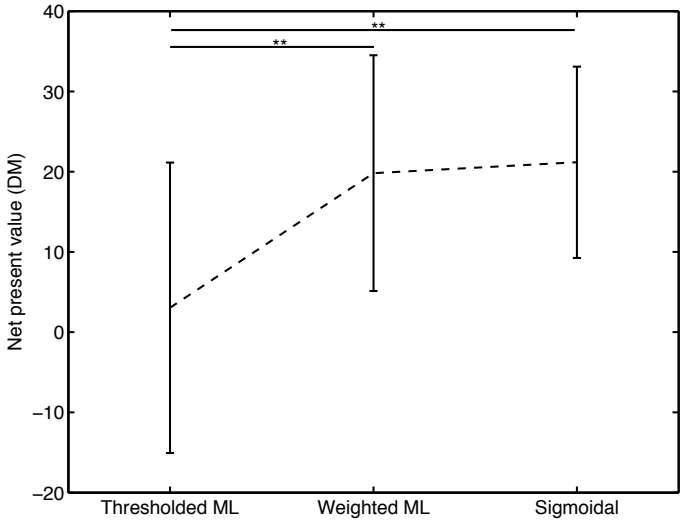


Figure 4: NPV per applicant: means and standard errors. Solid horizontal lines indicate statistical significance at the $p = 0.01$ level.

as whether the loan was repaid. This data set has been used previously to evaluate a novel profit-maximizing decision rule in the econometrics literature (Lieli and White, 2010).

Upon receiving a loan request, the bank decides either to grant the loan at a certain interest rate, or rather to invest the loan amount in a risk-free government bond. In this application, it is easier to work with benefits rather than costs; as such, the net present value (NPV), measured in Deutsche Marks (DM) of extending the loan must be compared with the NPV of rejecting the application, which is zero as outlined in Lieli and White (2010). The NPV of extending the loan depends on whether the loan is repaid, and thus the optimal decision rule takes into account the probability of repayment as well as the potential profit to be made on the loan. Thus, the aim of the evaluation is to predict, from an applicant's credit history and biographical information, the likelihood of the applicant repaying the loan which he/she is seeking.

In the framework described in the earlier sections, we have $c(y,\mathbf{x}) = y \cdot \pi(y,\mathbf{x})$, where $\pi(y,\mathbf{x})$ denotes the NPV associated with predicting $\hat{y} = +1$ when the truth is y (repayment: $y = +1$, default: $y = -1$). Please refer to Lieli and White (2010) for the precise mathematical relationship between the NPV π and the individual features in $\mathbf{x}$. In the evaluation, we used the $D = 5$ dimensional feature set chosen by Lieli and White (2010), as well as their proxies for the interest and risk-free government rates. Note that the "costs" are both example-dependent and endogenous to the problem. We conducted a leave-one-out cross-validation of the thresholded ML, weighted ML, and sigmoidal risk estimators on this $N = 1000$ example data set (MetaCost is not applicable to problems with example-dependent costs).

Figure 4 displays the mean NPV per applicant, along with standard errors. On average, the means obtained by the thresholded ML, weighted ML, and sigmoidal risk estimators are DM 3.0 , DM 19.8 , and DM 21.2 . The standard errors are given by DM 18.1 , DM 14.7, and DM 11.9, respectively (a negative value of NPV indicates an overall loss for the classification scheme). We

performed pairwise sign-tests of statistical significance to determine if performance differs significantly for the classifiers. A statistically significant improvement in NPV is achieved by both WML and sigmoidal risk estimators over the thresholded ML solution ($p < 0.01$). On the other hand, statistical significance cannot be established between WML and sigmoidal risk ($p \approx 0.9$). This empirical finding supports the analytical result of negative weighted log likelihood upper bounding empirical loss.

5. Discussion

It is interesting to point out the process which led to the findings presented in this paper. Initially, we were motivated by the observation that with a misspecified model, the direction of the minimum risk hyperplane is a function of the ratio of misclassification costs and the class priors. Since the ML approach is inherently to shift the minimum error hyperplane, we sought to develop an estimator which, given a ratio of misclassification costs, will find the direction required to minimize risk rather than maximizing likelihood. The expectation was that such an estimator would provide large performance gains over the ML approach. This led us to the development of the sigmoidal empirical risk minimizer. During the algorithm evaluation process, several findings emerged. Firstly, the search for the minimum risk hyperplane is non-trivial: regularization techniques proved to be necessary, particularly in the case of a limited training set. Moreover, both the existing and proposed cost-sensitive learning techniques yield the greatest benefits over thresholded ML when presented with large amounts of training data. When abundant data is available, the sigmoidal risk estimator typically outperforms all other methods, but weighted ML yields quite comparable values.

When the model set includes the true posterior, the threshold-shifted ML approach is optimal. This naturally brings us to the following question: why not employ a rich model set (for example, a multi-layer neural network), estimate its parameters using ML, and then shift the threshold by the log of the misclassification cost ratio? With an infinite amount of training data, we are sure to arrive at the lowest attainable risk. However, there are a few reasons why this procedure may not be desirable: a rich model set consists of many parameters, which in turn requires a large amount of training data to prevent over-fitting. From the so-called *structural risk minimization* principle, it is well-known that a simpler model set yields empirical risks that are closer to the true risk (Vapnik, 1998). Moreover, the optimality of the ML solution is not guaranteed for a finite amount of data. Thus, rates of convergence are key to determining the best approach.

In general, the choice of model complexity hinges upon several factors: the dimensionality of the feature space in relation to the number of available examples, the signal-to-noise ratio, and also the skew in class prevalence. For example, in applications involving a rare and expensive class, the key is to yield accurate decisions for this infrequent class. If the number of such examples is low, then even if the number of overall examples is high, a complex model will generally be undesirable. In other words, the effective sample size is closer to the number of costly examples than the entire sample size N. Consequently, the number of free parameters needs to be limited to prevent overfitting. The design issue in cost-sensitive learning is thus how best to use these few degrees of freedom: whether to "prioritize" correct decisions on the costly training examples, or rather to "spend" the degrees of freedom on achieving the best model fit.

The results with Gaussian data presented above appear to indicate that the sigmoidal risk minimizer tends to the true minimum risk model given enough data. However, the weighted ML estimator provides a tight upper bound on the sigmoidal empirical risk and thus this solution is not far

from optimal. Given that the negative weighted likelihood is convex, weighted ML thus becomes the preferred cost-sensitive technique.

Lastly, as seen in Figure 2, both the opportunity and the challenge in cost-sensitive learning lies in the ability to estimate the minimum risk model with limited data. Thus, the focus going forward should be on sparse (i.e., with few examples relative to the dimensionality of the feature space) inference of the minimum risk classifier.

6. Conclusion

This paper has elucidated the role of the specification of the model set in the problem of learning with asymmetric costs or class prevalences. It was shown that in the case of the model family including the true posterior, thresholding the ML solution is guaranteed to asymptotically minimize risk. In this case, cost-sensitive learning is synonymous with threshold adjustment. On the other hand, with a misspecified model, the risk minimizing solution is a function of the misclassification cost ratios, and thresholding the ML estimate is sub-optimal. A novel estimator based on a sigmoidal estimation of the empirical risk was presented and shown to outperform conventional techniques provided enough data; however, the negative weighted log likelihood was analytically and empirically shown to tightly upper bound the sigmoidal loss. Thus, we advocated for the weighted ML as the preferred cost-sensitive learning technique.

Acknowledgments

The authors are grateful to the editor and all of the anonymous reviewers for each providing numerous useful comments and suggestions. We would also like to thank Barak Pearlmutter for some insightful discussions.

This work was supported by The Defense Advanced Research Projects Agency (government contract no. NBCHC080029). We would also like to acknowledge support for this project from the National Science and Engineering Research Council of Canada (NSERC).

The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

Appendix A. MATLAB Code for Weighted Maximum Likelihood

Provided below is a basic MATLAB implementation of the weighted ML approach to cost-sensitive learning using iteratively reweighted weighted least squares. For a more detailed version as well as implementations of thresholded ML and sigmoidal risk minimization, please refer to `http://bme.ccny.cuny.edu/faculty/lparra/cost`.

```
function v = wml(x,y,c,lambda)
% x - N-by-D matrix of input samples \in (-inf,inf)
% y - N-by-1 vector of binary labels \in {0,1}
% c - N-by-1 vector of costs \in (0,inf)
% lambda - regularization parameter \in (0,inf) (defaults to 0)
% v - v(1:D) normal to separating hyperplane, v(D+1) threshold
% (c) Lucas C. Parra, Jacek P. Dmochowski
```

```
if nargin<4; lambda=0; end;
[N,D]=size(x);
s = std(x); x = x./repmat(s,[N 1]);
x = [x ones(N,1)];
v = zeros(D+1,1);
lambda = [0.5*lambda*ones(1,D) 0]';
while 1
    vold=v;
    mu = exp(x*v - log(1+exp(x*v)));
    w = ( mu.*(1-mu) ).*c;
    e = (y - mu).*c;
    grad = x'*e - lambda .* v;
    inc = inv(x'*(repmat(w,1,D+1).*x)+diag(lambda)) * grad;
    v = v + inc;
    if norm(vold) & subspace(v,vold)<10^-10, break, end;
end;
v(1:end-1) = v(1:end-1)./s';
```

References

A. Asuncion and D. J. Newman. Uci machine learning repository, 2007. URL `http://archive.ics.uci.edu/ml/`.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

N. V. Chawla and N. Japkowicz. Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explorations*, 6:2004, 2004.

P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

C. Drummond and R. C. Holte. Explicitly representing expected cost: An alternative to roc representation. In *In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 198–207. ACM Press, 2000.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, 2001.

M. Dudik and S. J. Phillips. Generative and discriminative learning with unknown labeling bias. *Advances in Neural information Processing Systems*, 21, 2009.

J. P. Egan. *Signal Detection Theory and ROC Analysis*. Academic Press, New York, NY, 1975.

C. Elkan. The foundations of cost-sensitive learning. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

G. Elliott and R. P. Lieli. Predicting binary outcomes. Technical report, 2007.

T. Fawcett. Roc graphs: notes and practical considerations for researchers. Technical Report HPL-2003-4, 2004.

R. Fletcher. *Practical Methods of Optimization*. Wiley, West Sussex, 2000.

A. Guerrero-Curieses, J. Cid-Sueiro, R. Alaiz-Rodriguez, and A. R. Figueras-Vidal. Local estimation of posterior class probabilities to minimize classification errors. *IEEE Transactions on Neural Networks*, 15:309–317, 2004.

J. L. Horowitz. A smoothed maximum score estimator for the binary response model. *Econometrica*, 60:505–531, 1992.

S. M. Kay. *Fundamentals of statistical signal processing: estimation theory*. 1993.

R. P. Lieli and H. White. The construction of empirical credit scoring rules based on maximization principles. *Journal of Econometrics*, 157:110–119, 2010.

Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46:191–202, 2002.

M. A. Maloof. Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 Workshop on Learning from Imbalanced Data Sets II*, 2003.

D. D. Margineantu. On class-probability estimates and cost-sensitive evaluation of classifiers. In *In Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML2000)*, 2000.

H. Masnadi-Shirazi and N. Vasconcelos. Risk minimization, probability elicitation, and cost-sensitive svms. In *Proceedings of the International Conference on Machine Learning*, pages 204–213. ACM Press, 2010.

P. McCullagh and J. A. Nelder. *Generalized Linear Models, 2nd ed*. Chapman and Hall, London, 1989.

L. Parra, C. Spence, A. Gerson, and P. Sajda. Recipes for the linear analysis of eeg. *NeuroImage*, 28:326–341, 2005.

L. Parra, C. Christoforou, A. Gerson, M. Dyrholm, A. Luo, M. Wagner, M. Philiastides, and P. Sajda. Spatio-temporal linear decoding of brain state: Application to performance augmentation in high-throughput tasks. *IEEE Signal Processing Magazine*, 25:95–115, 2008.

F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 43–48. AAAI Press, 1997.

H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Statistical Planning and Inference*, 90:227–244, 2000.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

H. White. Maximum likelihood estimation of misspecified models. *Econometrica*, 50:1–25, 1982.

B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 204–213. ACM Press, 2001.

B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *n Proceedings of the Third IEEE International Conference on Data Mining, ICDM 2003*., pages 435–442. IEEE, 2003.

Journal of Machine Learning Research 11 (2010) 3333-3369 Submitted 3/09; Revised 7/10; Published 12/10

Learning Instance-Specific Predictive Models

Shyam Visweswaran SHV3@PITT.EDU
Gregory F. Cooper GFC@PITT.EDU
Department of Biomedical Informatics
University of Pittsburgh
Pittsburgh, PA 15260, USA

Editor: Max Chickering

Abstract

This paper introduces a Bayesian algorithm for constructing predictive models from data that are optimized to predict a target variable well for a particular instance. This algorithm learns Markov blanket models, carries out Bayesian model averaging over a set of models to predict a target variable of the instance at hand, and employs an instance-specific heuristic to locate a set of suitable models to average over. We call this method the instance-specific Markov blanket (ISMB) algorithm. The ISMB algorithm was evaluated on 21 UCI data sets using five different performance measures and its performance was compared to that of several commonly used predictive algorithms, including nave Bayes, C4.5 decision tree, logistic regression, neural networks, *k*-Nearest Neighbor, Lazy Bayesian Rules, and AdaBoost. Over all the data sets, the ISMB algorithm performed better on average on all performance measures against all the comparison algorithms.

Keywords: instance-specific, Bayesian network, Markov blanket, Bayesian model averaging

1. Introduction

Prediction is a central problem in machine learning that involves inducing a model from a set of training instances that is then applied to future instances to predict a target variable of interest. Several commonly used predictive algorithms, such as logistic regression, neural networks, decision trees, and Bayesian networks, typically induce a single model from a training set of instances, with the intent of applying it to all future instances. We call such a model a *population-wide model* because it is intended to be applied to an entire population of future instances. A population-wide model is optimized to predict well on average when applied to expected future instances.

Recent research in machine learning has shown that inducing models that are specific to the particular features of a given instance can improve predictive performances (Gottrup et al., 2005). We call such a model an *instance-specific model* since it is constructed specifically for a particular instance (case). The structure and parameters of an instance-specific model are specialized to the particular features of an instance, so that it is optimized to predict especially well for that instance. The goal of inducing an instance-specific model is to obtain optimal prediction for the instance at hand. This is in contrast to the induction of a population-wide model where the goal is to obtain optimal predictive performance on average on all future instances.

There are several possible approaches for learning predictive models that are relevant to a single instance. One approach is to learn a model from a subset of the training data set that consists of instances that are similar in some way to the instance at hand. Another approach is to learn a

model from a subset of variables that are pertinent in some fashion to the instance at hand. A third approach, applicable to model averaging where a set of models is collectively used for prediction, is to identify a set of models that are most relevant to prediction for the instance at hand.

In this paper, we describe a new instance-specific method for learning predictive models that (1) uses Bayesian network models, (2) carries out Bayesian model averaging over a set of models to predict the target variable for the instance at hand, and (3) employs an instance-specific heuristic to identify a set of suitable models to average over. The remainder of this section gives a brief description of each of these characteristics.

Bayesian network (BN) models are probabilistic graphical models that provide a powerful formalism for representation, reasoning and learning under uncertainty (Pearl, 1988; Neapolitan, 2003). These graphical models are also referred to as probabilistic networks, belief networks or Bayesian belief networks. A BN model combines a graphical representation with numerical information to represent a probability distribution over a set of random variables in a domain. The graphical representation constitutes the BN structure, and it explicitly highlights the probabilistic independencies among the domain variables. The complementary numerical information constitutes the BN parameters, which quantify the probabilistic relationships among the variables. The instance-specific method that we describe in this paper uses Markov blanket models, which are a special type of BN models.

Typically, methods that learn predictive models from data, including those that learn BN models, perform model selection. In model selection a single model is selected that summarizes the data well; it is then used to make future predictions. However, given finite data, there is uncertainty in choosing one model to the exclusion of all others, and this can be especially problematic when the selected model is one of several distinct models that all summarize the data more or less equally well. A coherent approach to dealing with the uncertainty in model selection is Bayesian model averaging (BMA) (Hoeting et al., 1999). BMA is the standard Bayesian approach wherein the prediction is obtained from a weighted average of the predictions of a set of models, with more probable models influencing the prediction more than less probable ones. In practical situations, the number of models to be considered is enormous and averaging the predictions over all of them is infeasible. A pragmatic approach is to average over a few good models, termed *selective Bayesian model averaging*, which serves to approximate the prediction obtained from averaging over all models. The instance-specific method that we describe in this paper performs selective BMA over a set of models that have been selected in an instance-specific fashion.

The instance-specific method described here learns both the structure and parameters of BNs automatically from data. The instance-specific characteristic of the method is motivated by the intuition that in constructing predictive models, all the available information should be used including available knowledge of the features of the current instance. Specifically, the instance-specific method uses the features of the current instance to inform the BN learning algorithm to selectively average over models that differ considerably in their predictions for the target variable of the instance at hand. The differing predictions of the selected models are then combined to predict the target variable.

2. Characterization of Instance-Specific Models

Figure 1 illustrates the key difference between population-wide and instance-specific models: the instance-specific model is constructed from data in the training set, as well as, from the features

about the particular instance to which it will be applied. In contrast, the population-wide model is constructed only from data in the training set. Thus, intuitively, the additional information available to the instance-specific method can facilitate the induction of a model that provides better prediction for the instance at hand. In instance-specific modeling, different instances will potentially result in different models, because the instances contain potentially different values for the features.[1] The instance-specific models may differ in the variables included in the model (variable selection), in the interaction among the included variables (encoded in the structure of the model), and in the strength of the interaction (encoded in the parameters of the model). Another approach is to select a subset of the training data that are similar in their feature values to those of the instance at hand and learn the model from the subset. A generalization of this is to weight the instances in the training data set such that those that are more similar to the instance at hand are assigned greater weights than others, and then learn the model from the weighted data set. The following are two illustrative examples where instance-specific methods may perform better than population-wide methods.

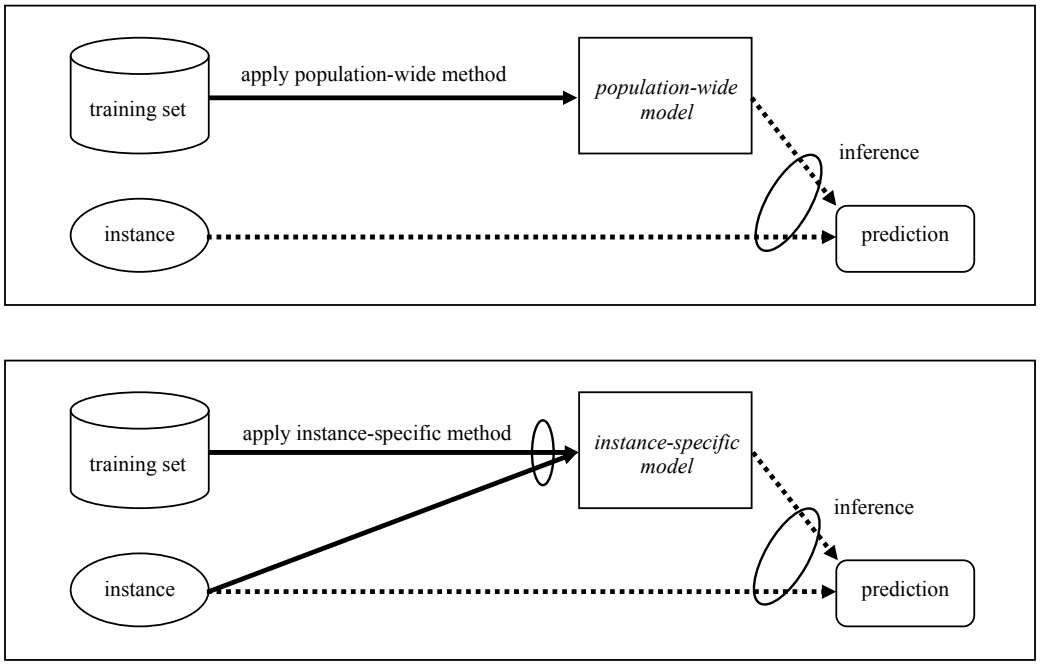


Figure 1: A general characterization of the induction of and inference in population-wide (top panel) and instance-specific (bottom panel) models. In the bottom panel, there is an extra arc from *instance* to *model*, because the structure and parameters of the model are influenced by the features of the instance at hand.

2.1 Variable Selection

Many model induction methods implicitly or explicitly perform variable selection, a process by which a subset of the domain variables is selected for inclusion in the model. For example, logistic

1. A feature is a variable-value pair, that is, a variable that has been assigned a value.

regression is often used with a stepwise variable selection process. An instance-specific version of logistic regression could, for example, select different variables for different instances being predicted, compared to the standard population-wide version that selects a single subset of variables. Consider a simple example where a gene G that has several alleles. Suppose that allele a1 is rare, and it is the only allele that predicts the development of disease D; indeed, it predicts D with high probability. For future instances, the aim is to predict $P(D|G)$. In a population-wide logistic regression model, G may not be included as a predictor (variable) of D, because in the vast majority of instances in the data set $G \neq a1$ and D is absent, and having G as a predictor would just increase the overall noise in predicting D. In contrast, if there is an instance at hand in which $G = a1$, then the training data may contain enough instances to indicate that D is highly likely. In this situation, G would be added as a predictor in an instance-specific model. Thus, for an instance in which $G = a1$, the typical population-wide logistic regression model would predict poorly, but an instance-specific model would predict well.

This idea can be extended to examples with more than one predictor, in which some predictors are characterized by having particular values that are relatively rare but strongly predictive for the outcome. A population-wide model tends to include only those predictors that on average provide the best predictive performance. In contrast, an instance-specific model will potentially include predictors that are highly predictive for the particular instance at hand; such predictors may be different from those included in the population-wide model.

2.2 Decision Theoretic Comparison of Population-Wide and Instance-Specific Models

We first introduce some notation and definitions and then compare population-wide with instance-specific models in decision theoretic terms. Capital letters like X, Z, denote random variables and corresponding lower case letters, x, z, denote specific values assigned to them. A feature is a specification of a variable and its value. Thus, $X = x$ is a feature that specifies that variable X is assigned the value x. Bold upper case letters, such as $\mathbf{X}$, $\mathbf{Z}$, represent sets of variables or random vectors, and their realization is denoted by the corresponding bold lower case letters, $\mathbf{x}$, $\mathbf{z}$. A feature vector is a list of features. Thus, $\mathbf{X} = \mathbf{x}$ is a feature vector that specifies that the variables in $\mathbf{X}$ have the values given by $\mathbf{x}$. In addition, Z denotes the target variable (class variable) being predicted, $\mathbf{X}$ denotes the set of predictor variables, M denotes a model (including both its structure and parameters), D denotes the training data set, $C^i =< \mathbf{X}^i, Z^i >$ denotes a generic training instance in D and $C^t =< \mathbf{X}^t, Z^t >$ denotes a generic test instance that is not in D. A test instance t is one in which the unknown value of the target variable Z^t is to be predicted from the known values of the predictors $\mathbf{X}^t$ and the known values of $< \mathbf{X}^i, Z^i >$ of a set of training instances.

A probabilistic *model* is a family of probability distributions indexed by a set of parameters. *Model selection* refers to the problem of using data to select one model from a set of models under consideration (Wasserman, 2000). The process of selecting a model typically involves model class selection (e.g., logistic regression, BN), variable selection, and parameter estimation. *Model averaging* refers to the process of estimating some quantity (e.g., prediction of the value of a target variable) under each of the models under consideration and obtaining a weighted average of their estimates (Wasserman, 2000).

Model selection can be done using either non-Bayesian or Bayesian approaches. Non-Bayesian methods of model selection include choosing among competing models by maximizing the likelihood, by maximizing a penalized version of the likelihood or by maximizing some measure of

interest (e.g., accuracy) using cross-validation. Use of multiple models to improve performance can also be done using either non-Bayesian or Bayesian approaches. Ensemble techniques such as bagging and boosting are non-Bayesian approaches that combine multiple models to create a new better performing model. In both bagging and boosting, the data are resampled several times, a model is constructed from each sample, and the predictions of the individual models are combined to obtain the final prediction. In the non-Bayesian approach, the heuristics used in model selection and model combination are typically different. In contrast, the Bayesian approach to model selection and model combination both involve computing the posterior probability of each model under consideration. In Bayesian model selection the single model found that has the highest posterior probability is chosen. The Bayesian model combination technique is called model averaging where the combined prediction is the weighted average of the individual predictions of the models with the model posterior probabilities comprising the weights.

When the goal is prediction of future data or future values of the target variable, BMA is preferred, since it suitably incorporates the uncertainty about the identity of the true model. However, sometimes interest is focused on a single model. For example, a single model may be useful for providing insight into the relationships among the domain variables or can be used as a computationally less expensive method for prediction. In such cases, Bayesian model selection maybe preferred to BMA. However, the optimal Bayesian approach is to perform model averaging, and thus, model selection is at best an approximation to model averaging.

Population-wide model selection and instance-specific model selection are characterized in decision theoretic terms as follows. In this paper, all conditional probabilities have a conditioning event K, which represents background knowledge and which we will leave implicit for notational simplicity. Given training data D and a generic test instance $< \mathbf{X}^t, Z^t >$, the *optimal population-wide model* is:

$$\arg\max_{M} \left\{ \sum_{\mathbf{X}^t} U\left[P(Z^t|\mathbf{X}^t,D), P(Z^t|\mathbf{X}^t,M)\right] P(\mathbf{X}^t|D) \right\} \tag{1}$$

where the utility function U gives the utility of approximating the *Bayes optimal estimate* $P(Z^t|\mathbf{X}^t,D)$ with the estimate $P(Z^t|\mathbf{X}^t,M)$ obtained from model M. For a model M, Expression 1 considers all possible instantiations of $\mathbf{X}^t$ and for each instantiation computes the utility of estimating $P(Z^t|\mathbf{X}^t,D)$ with the specific model estimate $P(Z^t|\mathbf{X}^t,M)$, and weights that utility by the posterior probability of that instantiation. The maximization is over the models M in a given model space.

The *Bayes optimal estimate* $P(Z^t|\mathbf{X}^t,D)$ in Expression 1 is obtained by combining the estimates of all models (in a given model space) weighted by their posterior probabilities:

$$P(Z^t|\mathbf{X}^t,D) = \int_M P(Z^t|\mathbf{X}^t,M)P(M|D)dM. \tag{2}$$

The term $P(\mathbf{X}^t|D)$ in Expression 1 is given by:

$$P(\mathbf{X}^t|D) = \int_M P(\mathbf{X}^t|M)P(M|D)dM. \tag{3}$$

The *optimal instance-specific model* for estimating Z^t is the one that maximizes the following:

$$\arg\max_{M} \left\{ U\left[P(Z^t|\mathbf{x}^t,D), P(Z^t|\mathbf{x}^t,M)\right] \right\}, \tag{4}$$

where $\mathbf{x}^t$ are the values of the predictors of the test instance $\mathbf{X}^t$ for which the target variable Z^t is to be predicted. The *Bayes optimal instance-specific prediction* $P(Z^t|\mathbf{X}^t, D)$ is derived using Equation 2, for the special case in which $\mathbf{X}^t = \mathbf{x}^t$, as follows:

$$P(Z^t|\mathbf{x}^t, D) = \int_M P(Z^t|\mathbf{x}^t, M)P(M|D)dM.$$

The difference between the population-wide and the instance-specific model selection can be noted by comparing Expressions 1 and 4. Expression 1 for the population-wide model selects the model that on average will have the greatest utility. Expression 4 for the instance-specific model, however, selects the model that will have the greatest utility for the specific instance $\mathbf{X}^t = \mathbf{x}^t$. For predicting Z^t given instance $\mathbf{X}^t = \mathbf{x}^t$, application of the model selected using Expression 1 can never have an expected utility greater than the application of the model selected using Expression 4. This observation provides support for developing instance-specific models.

Equations 2 and 3 carry out BMA over all models in some specified model space. Expressions 1 and 4 include Equation 2; thus, these expressions for population-wide and instance-specific model selection, respectively, are theoretical ideals. Moreover, Equation 2 is the Bayes optimal prediction of Z^t. Thus, in order to do optimal model selection, the optimal prediction obtained from BMA must already be known.

Model selection, even if performed optimally, ignores the uncertainty inherent in choosing a single model based on limited data. BMA is a normative approach for dealing with the uncertainty in model selection. Such averaging is primarily useful when no single model in the model space under consideration has a dominant posterior probability. However, since the number of models in practically useful model spaces is enormous, *exact BMA*, where the averaging is done over the entire model space, is usually not feasible. That is, it is usually not computationally feasible to solve for the exact solution given by Equation 2. In such cases, *selective BMA* is typically performed, where the averaging is done over a selected subset of models.

BMA has been shown to improve predictive performance, and several examples of significant decrease in prediction errors with the use of BMA are described by Hoeting et al. (1999). However, in other cases BMA has not proved to be better than ensemble techniques. For example, uniform averaging was shown by Cerquides and Mantaras (2005) to have better classification performance than BMA for one dependence estimators. This may be because, as Minka (2002) points out, BMA is better described as a method for 'soft model selection' rather than a technique for model combination.

3. Related Work

There exists a vast literature in machine learning, data mining and pattern recognition that is concerned with the problem of predictive modeling and supervised learning. We briefly describe some of the aspects of the similarity-based methods and instance-specific methods because these methods are most closely relevant to the present paper. Similarity-based methods are characterized by the use of a similarity (or distance) measure necessary for measuring the similarity between instances. Instance-specific methods, on the other hand, learn an explicit model or models from the training instances that are then applied to the test instance. The induction of a model or set of models are influenced by the values of the features of the test instance, and a similarity measure is not used.

3.1 Similarity-Based Methods

These methods are also known as memory-based, case-based, instance-based, or exemplar-based learners. They (1) use a similarity or a distance measure, (2) defer most of the processing until a test instance is encountered, (3) combine the training instances in some fashion to predict the target variable in the test instance, and (4) discard the answer and any intermediate results after the prediction. Typically, no explicit model is induced from the training instances at the time of prediction (Aha, 1998). The similarity measure evaluates the similarity between the test instance and the training instances and selects the appropriate training instances and their relative weights in response to the test instance (Zhang et al., 1997). The selected training instances can be equally weighted or weighted according to their similarity to the test instance. To predict the target variable in the test instance, the values of the target variable in the selected training instances are combined in some simple fashion such as majority vote, simple numerical average or fitted with a polynomial.

The *nearest-neighbor technique* is the canonical similarity-based method. When a test instance is encountered, the training instance that is most similar to the test instance is located and its target value is returned as the prediction (Cover and Hart, 1967). A straight-forward extension to the nearest-neighbor technique is the *k-Nearest Neighbor*(*k*NN) method. For a test instance, this method selects the k most similar training instances and either averages or takes a majority vote of their target values. Another extension is the distance-weighted k-Nearest Neighbor method. This weights the contribution of each of the k most similar training instances according to its similarity to the test instance, assigning greater weights to more similar instances (Dasarathy, 1991). A further extension is locally weighted regression that selects instances similar to the test instance, weights them according to their similarity, and performs regression to predict the target (Atkeson et al., 1997).

One drawback of the similarity-based methods is that they may perform poorly when predictors are redundant, irrelevant or noisy. To make the similarity metrics more robust, variable selection and variable weighting have been employed.

3.2 Instance-Specific Methods

Instance-specific methods are model-based methods that take advantage of the features in the test instance while inducing a model. Such methods are not as reliant on a similarity measure, if they use one at all, as the similarity-based methods.

Friedman et al. (1996) describe one such algorithm called LazyDT that searches for the best CART-like decision tree for a test instance. As implemented by the authors, LazyDT did not perform pruning and processed only nominal variables. The algorithm was compared to ID3 and C4.5 (standard population-wide methods for inducing decision trees), each with and without pruning. When evaluated on 28 data sets from the UCI Machine Learning repository, LazyDT generally out-performed both ID3 and C4.5 without pruning and performed slightly better than C4.5 with pruning.

Ting et al. (1999) developed a framework for inducing rules in a lazy fashion that are tailored to the features of the test instance. Zheng and Webb (2000) describe an implementation of this framework called the Lazy Bayesian Rules (LBR) learner that induces a rule tailored to the features of the test instance that is then used to classify it. A LBR rule consists of (1) a conjunction of the features (variable-value pairs) present in the test instance as the antecedent, and (2) a local naive Bayes classifier as the consequent. The structure of the local naive Bayes classifier consists of

the target variable as the parent of all other variables that do not appear in the antecedent, and the parameters of the classifier are estimated from those training instances that satisfy the antecedent. A greedy step-forward search selects the optimal LBR rule for a test instance to be classified. In particular, each predictor is added to the antecedent of the current best rule and evaluated for whether it reduces the overall error rate on the training set that is estimated by cross-validation. The predictor that most reduces the overall error rate is added to the antecedent and removed from the consequent, and the search continues; if no single predictor move can decrease the current error rate, then the search halts and the current rule is applied to predict the outcome for the test instance. LBR is an example of an instance-specific method that uses feature information available in the test instance to direct the search for a suitable model in the model space.

The performance of LBR was evaluated by Zheng and Webb (2000) on 29 data sets from the UCI Machine Learning repository and compared to that of seven algorithms: a nave Bayes classifier (NB), a decision tree algorithm (C4.5), a Bayesian tree learning algorithm (NBTree) (Kohavi, 1996), a constructive Bayesian classifier that replaces single variables with new variables constructed from Cartesian products of existing nominal variables (BSEJ) (Pazzani, 1998), a selective naive Bayes classifier that deletes irrelevant variables using Backward Sequential Elimination (BSE) (Pazzani, 1995), and LazyDT, which is described above. Based on ten three-fold cross validation trials (for a total of 30 trials), LBR achieved the lowest average error rate across the 29 data sets. The average relative error reduction of LBR over NB, C4.5, NBTree, BSEJ, BSE and LazyDT were 9%, 10%, 2%, 3%, 5% and 16% respectively. LBR performed significantly better than all other algorithms except BSE; compared to BSE its performance was better but not statistically significantly so.

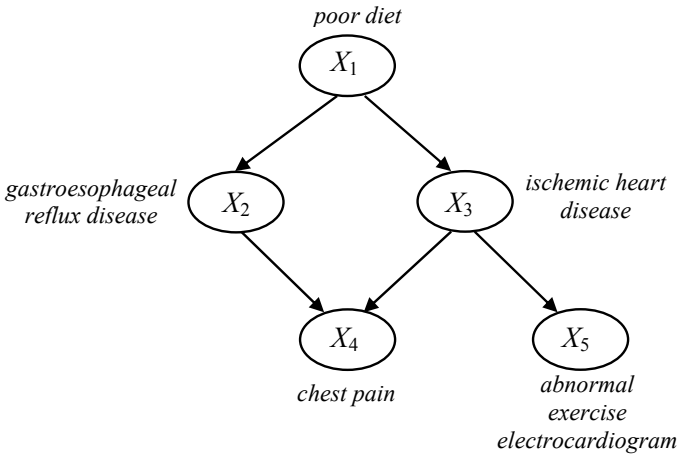
The instance-specific algorithms like LazyDT and LBR have limitations in that they can process only discrete variables, and continuous variables have to be discretized. Also, they are computationally more intensive than many other learning algorithms. However, they have been shown to have better accuracy than several of the population-wide methods.

4. Bayesian Networks

A Bayesian network (BN) is a probabilistic model that combines a graphical representation (the BN structure) with quantitative information (the BN parameterization) to represent a joint probability distribution over a set of random variables (Pearl, 1988; Neapolitan, 2003). More specifically, a BN model M representing the set of random variables $\mathbf{X}$ for some domain consists of a pair G, θ_G. The first component G is a directed acyclic graph (DAG) that contains a node for every variable in $\mathbf{X}$ and an arc between a pair of nodes if the corresponding variables are directly probabilistically dependent. Conversely, the absence of an arc between a pair of nodes denotes probabilistic independence between the corresponding variables. In this paper, the terms variable and node are used interchangeably in the context of random variables being modeled by nodes in a BN. Thus, a variable X_i in the domain of interest is represented by a node labeled X_i in the BN graph. Note that the phrase *BN structure* refers only to the graphical structure G, while the term BN (model) refers to both the structure G and a corresponding set of parameters θ_G.

The terminology of kinship is used to denote various relationships among nodes in a graph. These kinship relations are defined along the direction of the arcs. Predecessors of a node X_i in G, both immediate and remote, are called the *ancestors* of X_i. In particular, the immediate predecessors of X_i are called the *parents* of X_i. The set of parents of X_i in G is denoted by $\mathbf{Pa}(X_i, G)$ or more simply as $\mathbf{Pa}_i$ when the BN structure is obvious from the context. In a similar fashion, successors of

X_i in G, both immediate and remote, are called the *descendants* of X_i, and the immediate successors are called the *children* of X_i. A node X_j is termed a *spouse* of X_i if X_j is a parent of a child of X_i. The set of nodes consisting of a node X_i and its parents is called the *family* of X_i. Figure 2 gives an illustrative example of a simple hypothetical BN, where the top panel shows the graphical component G of the BN. In the figure, the variable *poor diet* is a parent of the variable *ischemic heart disease* as well as a parent of the variable *gastroesophageal reflux disease*. The variable *chest pain* is a child of the variable *lung cancer* as well as a child of the variable *gastroesophageal reflux disease*, and the variables *ischemic heart disease* and *abnormal electrocardiogram* are descendants of the variable *poor diet*.



Node X_1	$P(X_1 = \text{F}) = 0.70$	$P(X_1 = \text{T}) = 0.30$
Node X_2	$P(X_2 = \text{F} \mid X_1 = \text{F}) = 0.97$	$P(X_2 = \text{T} \mid X_1 = \text{F}) = 0.03$
	$P(X_2 = \text{F} \mid X_1 = \text{T}) = 0.96$	$P(X_2 = \text{T} \mid X_1 = \text{T}) = 0.04$
Node X_3	$P(X_3 = \text{F} \mid X_1 = \text{F}) = 0.94$	$P(X_3 = \text{T} \mid X_1 = \text{F}) = 0.06$
	$P(X_3 = \text{F} \mid X_1 = \text{T}) = 0.96$	$P(X_3 = \text{T} \mid X_1 = \text{T}) = 0.08$
Node X_4	$P(X_4 = \text{F} \mid X_2 = \text{F}, X_3 = \text{F}) = 0.90$	$P(X_4 = \text{T} \mid X_2 = \text{F}, X_3 = \text{F}) = 0.10$
	$P(X_4 = \text{F} \mid X_2 = \text{F}, X_3 = \text{T}) = 0.40$	$P(X_4 = \text{T} \mid X_2 = \text{F}, X_3 = \text{T}) = 0.60$
	$P(X_4 = \text{F} \mid X_2 = \text{T}, X_3 = \text{F}) = 0.50$	$P(X_4 = \text{T} \mid X_2 = \text{T}, X_3 = \text{F}) = 0.50$
	$P(X_4 = \text{F} \mid X_2 = \text{T}, X_3 = \text{T}) = 0.25$	$P(X_4 = \text{T} \mid X_2 = \text{T}, X_3 = \text{T}) = 0.75$
Node X_5	$P(X_5 = \text{F} \mid X_3 = \text{F}) = 0.80$	$P(X_5 = \text{T} \mid X_3 = \text{F}) = 0.20$
	$P(X_5 = \text{F} \mid X_3 = \text{T}) = 0.25$	$P(X_5 = \text{T} \mid X_3 = \text{T}) = 0.75$

Figure 2: A simple hypothetical Bayesian network for a medical domain. All the nodes represent binary variables, taking values in the domain T, F where T stands for True and F for False. The graph at the top represents the Bayesian network structure. Associated with each variable (node) is a conditional probability table representing the probability of each variable's value conditioned on its parent set. (Note: these probabilities are for illustration only; they are not intended to reflect frequency of events in any actual patient population.)

The second component θ_G represents the parameterization of the probability distribution over the space of possible instantiations of $\mathbf{X}$ and is a set of local probabilistic models that encode quantitatively the nature of dependence of each variable on its parents. For each node X_i there is a probability distribution (that may be discrete or continuous) defined on that node for each state of its parents. The set of all the probability distributions associated with all the nodes comprises the complete parameterization of the BN. The bottom panel in Figure 2 gives an example of a set of parameters for G. Taken together, the top and bottom panels in Figure 2 provide a fully specified structural and quantitative representation for the BN.

4.1 Markov Blanket

The *Markov blanket* of a variable X_i, denoted by $\mathrm{MB}(X_i)$, defines a set of variables such that conditioned on $\mathrm{MB}(X_i)$ is conditionally independent of all variables given $\mathrm{MB}(X_i)$ for joint probability distributions consistent with BN in which $\mathrm{MB}(X_i)$ appears (Pearl, 1988). The minimal Markov blanket of a node X_i, which is sometimes called its Markov boundary, consists of the parents, children, and children's parents of X_i. In this paper, we refer to the minimal Markov blanket as the Markov blanket (MB). This entails that the variables in $\mathrm{MB}(X_i)$ are sufficient to determine the probability distribution of X_i. Since d-separation is applied to the graphical structure of a BN to identify all conditional independence relations, it can also be applied to identify the MB of a node in a BN. The MB of a node X_i consists of its parents, its children, and its children's parents and is illustrated in Figure 3. The parents and children of X_i are directly connected to it. In addition, the spouses are also included in the MB, because of the phenomenon of explaining away which refers to the observation that when a child node is instantiated its parents in general are statistically dependent. Analogous to BNs, the *MB structure* refers only to the graphical structure while the MB (model) refers to both the structure and a corresponding set of parameters.

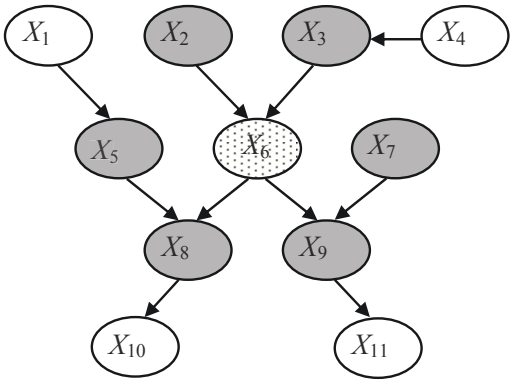


Figure 3: Example of a Markov blanket. The Markov blanket of the node X_6 (shown stippled) comprises the set of parents, children and spouses of the node and is indicated by the shaded nodes. The nodes in the Markov blanket include X_2 and X_3 as parents, X_8 and X_9 as children, and X_5 and X_7 as spouses of X_6. X_1, X_4, X_{10} and X_{11} are not in the Markov blanket of X_6.

The MB of a node is noteworthy because it identifies all the variables that shield the node from the rest of the network. In particular, when interest centers on the distribution of a specific target

node, as is the case in classification, the structure and parameters of only the MB of the target node need be learned.

4.2 Markov Blanket Algorithms

Many approaches for learning general BNs as well as for learning MBs from data have been described in the literature. Here we briefly review algorithms that learn MB classifiers. One of the earliest described MB learning algorithms is the Grow-Shrink (GS) Markov blanket algorithm that orders the variables according to the strength of association with the target and uses conditional independence tests to find a reduced set of variables estimated to be the MB (Margaritis and Thrun, 1999). Madden (2002a,b) described the Markov Blanket Bayesian Classifier (MBBC) algorithm that constructs an approximate MB classifier using a Bayesian score for evaluating the network. The algorithm consists of three steps: the first step identifies a set of direct parents and children of the target, the second step identifies a set of parents of the children, and the third step identifies dependencies among the children. The MBBC was competitive in terms of speed and accuracy relative to Nave Bayes, Tree-Augmented Nave Bayes and general Bayesian networks, when evaluated on a large set of UCI data sets.

Several MB algorithms have been developed in the context of variable selection and learning local causal structures around target variables of interest. Koller and Sahami (1996) showed that the optimal set of variables to predict a target is its MB. They proposed a heuristic entropy-based procedure (commonly referred to as the KS algorithm) that assumes that the target influences the predictor variables and that the variables most strongly associated with the target are in its MB. The KS algorithm was not guaranteed to succeed. Tsamardinos and Aliferis (2003) showed that for faithful distributions, the MB of a target variable is exactly the set of strongly relevant features, and developed the Incremental Association Markov Blanket (IAMB) to identify it. This algorithm has two stages: a growing phase that adds potential predictor variables to the MB and a shrinking phase that removes the false positives that were added in the first phase. Based on the faithfulness assumption, Tsamardinos et al. (2006) later developed the Min-Max Markov Blanket algorithm (MMMB) that first identifies the direct parents and children of the target and then parents of the children using conditional independence tests. A comparison of the efficiency of several MB learning algorithms are provided by Fu and Desmarais (2008). A recent comprehensive overview of MB methods of classification and the local structure learning is provided by Aliferis et al. (2010a,b).

Several methods for averaging over BNs for prediction or classification have been described in the literature, including Dash and Cooper (2002), Dash and Cooper (2004) and Hwang and Zhang (2005). In prior work, we developed a lazy instance-specific algorithm that performs BMA over LBR models (Visweswaran and Cooper, 2004) and showed that it had better classification performance than did model selection. However, to our knowledge, averaging over MBs has not been described in the literature.

5. The Instance-Specific Markov Blanket (ISMB) Algorithm

The goal of the instance-specific Markov blanket (ISMB) algorithm is to predict well a discrete target variable of interest. Relative to some model space, BMA is the optimal method for making predictions in the sense that it achieves the lowest expected error rate in predicting the outcomes of future instances. Such Bayes optimal predictions involve averaging over all models in the model space which is usually computationally intractable. One approach, termed *selective model averag-*

ing, has been to approximate the Bayes optimal prediction by averaging over a subset of the possible models and has been shown to improve predictive performance (Hoeting et al., 1999; Raftery et al., 1997; Madigan and Raftery, 1994). The ISMB algorithm performs selective model averaging and uses a novel heuristic search method to select the models over which averaging is done. The instance-specific characteristic of the algorithm arises from the observation that the search heuristic is sensitive to the features of the particular instance at hand.

The model space employed by the ISMB algorithm is the space of BNs over the domain variables. In particular, the algorithm considers only MBs of the target node, since a MB is sufficient for predicting the target variable. The remainder of this section describes the ISMB algorithm in terms of the (1) model space, (2) scoring functions including parameter and structure priors, and (3) the search procedure for exploring the space of models. The current version of the algorithm handles discrete variables.

5.1 Model Space

As mentioned above, the ISMB algorithm learns MBs of the target variable rather than entire BNs over all the variables. Typically, BN structure learning algorithms that learn from data induce a BN structure over all the variables in the domain. The MB of the target variable can be extracted from the learned BN structure by ignoring those nodes and their relations that are not members of the MB. The ISMB algorithm modifies the typical BN structure learning algorithm to learn only MBs of the target node of interest, by using a set of operators that generate only the MB structures of the target variable.

The ISMB algorithm is a search-and-score method that searches in the space of possible MB structures. Both, the BN structure learning algorithms and the MB structure learning algorithm used by ISMB, search in a space of structures that is exponential in the number of domain variables. Though the number of MB structures grows more slowly than the number of BN structures with the number of domain variables, the number of MB structures is still exponential in the number of variables (Visweswaran and Cooper, 2009). Thus, exhaustive search in this space is infeasible for domains containing more than a few variables and heuristic search is appropriate.

5.2 Instance-Specific Bayesian Model Averaging

The objective of the ISMB algorithm is to derive the posterior distribution $P(Z^t|,\mathbf{x}^t,D)$ for the target variable Z^t in the instance at hand, given the values of the other variables $\mathbf{X}^t = \mathbf{x}^t$ and the training data D. The ideal computation of the posterior distribution $P(Z^t|,\mathbf{x}^t,D)$ by BMA is as follows:

$$P(Z^t|\mathbf{x}^t,D) = \sum_{G \in M} P(Z^t|\mathbf{x}^t,G,D)P(G|D), \tag{5}$$

where the sum is taken over *all* MB structures G in the model space M. The first term on the right hand side, $P(Z^t|\mathbf{x}^t,G,D)$, is the probability $P(Z^t|\mathbf{x}^t)$ computed with a MB that has structure G and parameters $\hat{\theta}_G$ that are given by Equation 6 below. This parameterization of G produces predictions equivalent to those obtained by integrating over all the possible parameterizations for G. The second term, $P(G|D)$, is the posterior probability of the MB structure G given the training data D. In essence, Equation 5 states that a conditional probability of interest $P(Z^t|\mathbf{x}^t)$ is derived by taking a weighted average of that probability over all MB structures, where the weight associated

with a MB structure is the probability of that MB structure given the data. In general, $P(Z^t|\mathbf{x}^t)$ will have different values for the different sets of models over which the averaging is carried out.

5.3 Inference in Markov Blankets

Computing $P(Z^t|\mathbf{x}^t, G, D)$ in Equation 5 involves doing inference in the MB with a specified structure G. First, the parameters of the MB structure G are estimated using Bayesian parameters as given by the following expression (Cooper and Herskovits, 1992; Heckerman, 1999):

$$P(X_i = k|\mathbf{Pa}_i = j) \equiv \hat{\theta}_{ijk} = \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}} \tag{6}$$

where (1) N_{ijk} is the number of instances in data set D in which $X_i = k$ and the parents of X_i have the state denoted by j, (2) $N_{ij} = \sum_k N_{ijk}$, (3) α_{ijk} is a parameter prior that can be interpreted as belief equivalent to having previously (prior to obtaining D seen α_{ijk} instances in which $X_i = k$ and the parents of X_i have the state denoted by j, and (4) $\alpha_{ij} = \sum_k \alpha_{ijk}$. The $\hat{\theta}_{ijk}$ in Equation 6 represent the expected value of the probabilities that are derived by integrating over all possible parameter values. For the ISMB algorithm we set α_{ijk} to 1 for all i, j, and k, as a simple non-informative parameter prior (Cooper and Herskovits, 1992). Next, the parameterized MB is used to compute the distribution over the target variable Z^t of the instance at hand given the values $\mathbf{x}^t$ of the remaining variables in the MB by applying standard BN inference (Neapolitan, 2003).

5.4 Bayesian Scoring of Markov Blankets

In the Bayesian approach, the scoring function is based on the posterior probability $P(G|D)$ of the BN structure G given data D. This is the second term on the right hand side in Equation 3. The Bayesian approach treats the structure and parameters as uncertain quantities and incorporates prior distributions for both. The specification of the structure prior $P(G)$ assigns prior probabilities for the different MB structures. Application of Bayes rule gives:

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)}. \tag{7}$$

Since the denominator $P(D)$ does not vary with the structure, it simply acts as a normalizing factor that does not distinguish between different structures. Dropping the denominator yields the following Bayesian score:

$$score(G;D) = P(D|G)P(G). \tag{8}$$

The second term on the right in Equation 8 is the prior over structures, while the first term is the marginal likelihood (also know as the integrated likelihood or evidence) which measures the goodness of fit of the given structure to the data. The marginal likelihood is computed as follows:

$$P(D|G) = \int_{\theta_G} P(D|\theta_G, G)P(\theta_G|G)d\theta_G, \tag{9}$$

where $P(D|\theta_G, G)$ is the likelihood of the data given the BN (G, θ_G) and $P(\theta_G|G)$ is the specified prior distribution over the possible parameter values for the network structure G. Intuitively, the marginal likelihood measures the goodness of fit of the structure over all possible values of its parameters. Note that the marginal likelihood is distinct from the maximum likelihood, though both

are computed from the same function: the likelihood of the data given the structure. The maximum likelihood is the maximum value of this function while the marginal likelihood is the integrated (or the average) value of this function with the integration being carried out with respect to the prior $P(\theta_G|G)$.

Equation 9 can be evaluated analytically when the following assumptions hold: (1) the variables are discrete and the data D is a multinomial random sample with no missing values; (2) global parameter independence, that is, the parameters associated with each variable are independent; (3) local parameter independence, that is, the parameters associated with each parent state of a variable are independent; and (4) the parameters' prior distribution is Dirichlet. Under the above assumptions, the closed form for $P(D|G)$ is given by (Cooper and Herskovits, 1992; Heckerman, 1999):

$$P(D|G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})}, \tag{10}$$

where Γ denotes the Gamma function,n is the number of variables in G, q_i is the number of joint states of the parents of variable X_i that occur in D, r_i is the number of states of X_i that occur in D, and $\alpha_{ij} = \sum_k \alpha_{ijk}$. Also, as previously described, N_{ijk} is the number of instances in the data where node i has value k and the parents of i have the state denoted by j, and $N_{ij} = \sum_k N_{ijk}$.

The Bayesian score in Equation 7 incorporates both structure and parameter priors. The term $P(G)$ represents the structure prior and is the prior probability assigned to the BN structure G. For the ISMB algorithm, a uniform prior belief over all G is assumed which makes the term $P(G)$ a constant. Thus, $P(G|D)$ is equal to $P(D|G)$ up to a proportionality constant and the Bayesian score for $P(G)$ is defined simply as the marginal likelihood as follows:

$$score(G;D) = P(D|G) \propto P(G|D). \tag{11}$$

The parameter priors are incorporated in the marginal likelihood $P(D|G)$ as is obvious from the presence of the alpha terms in Equation 10. For the ISMB algorithm we set α_{ijk} to 1 for all i, j, and k in Equation 10, as a simple non-informative parameter prior, as mentioned in the previous section.

5.5 Selective Bayesian Model Averaging

Since Equation 5 sums over a very large number of MB structures, it is not feasible to compute it exactly. Hence, complete model averaging given by Equation 5 is approximated with selective model averaging, and heuristic search (described in the next section) is used to sample the model space. For a set R of MB structures that have been chosen from the model space by heuristic search, selective model averaging estimates $P(Z^t|\mathbf{x}^t, G)$ as:

$$P(Z^t|\mathbf{x}^t, D) \cong \sum_{G \in R} P(Z^t|\mathbf{x}^t, G, D) \frac{P(G|D)}{\sum_{G' \in R} P(G'|D)}. \tag{12}$$

Substituting Equation 11 into Equation 12, we obtain:

$$P(Z^t|\mathbf{x}^t, D) \cong \sum_{G \in R} P(Z^t|\mathbf{x}^t, G, D) \frac{score(G;D)}{\sum_{G' \in R} score(G';D)}. \tag{13}$$

The ISMB algorithm performs selective model averaging and seeks to locate a good set of models over which averaging is carried out.

5.6 Instance-Specific Search

The ISMB algorithm uses a two-phase search to sample the space of MB structures. The first phase (phase 1) ignores the evidence $\mathbf{x}^t$ from the instance at hand, while searching for MB structures that best fit the training data. The second phase (phase 2) continues to add to the set of MB structures obtained from phase 1, but now searches for MB structures that have the greatest impact on the prediction of Z^t for the instance at hand. We now describe in greater detail the two phases of the search.

Phase 1 uses *greedy hill-climbing search* and accumulates the best model discovered at each iteration of the search into a set R. At each iteration of the search, successor models are generated from the current best model; the best of the successor models is added to R *only if* this model is better than current best model; and the remaining successor models are discarded. Since, no backtracking is performed, phase 1 search terminates in a local maximum.

Phase 2 uses *best-first search* and adds the best model discovered at each iteration of the search to the set R. Unlike greedy hill-climbing search, best-first search holds models that have not been expanded (i.e., whose successors have not be generated) in a *priority queue* Q. At each iteration of the search, successor models are generated from the current best model and added to Q; after an iteration the best model from Q is added to R *even if* this model is not better than the current best model in R. Phase 2 search terminates when a user set criterion is satisfied. Since, the number of successor models that are generated can be quite large, the priority queue Q is limited to a capacity of at most w models. Thus, if Q already contains w models, addition of a new model to it leads to removal of the worst model from it. The queue allows the algorithm to keep in memory up to the best w scoring models found so far, and it facilitates limited backtracking to escape local maxima.

5.7 Search Operators and Scores

The operators used by the ISMB algorithm to traverse the space of MB structures are the same as those used in standard BN structure learning with minor modifications. The standard BN structure learning operators are (1) add an arc between two nodes if one does not exist, (2) delete an existing arc, and (3) reverse an existing arc, with the constraint that an operation is allowed only if it generates a legal BN structure (Neapolitan, 2003). This constraint simply implies that the graph of the generated BN structure be a DAG. A similar constraint is applicable to the generation of MB structures, namely, that an operation is considered valid if it produces a legal MB structure of the target node. This constraint entails that some of the operations be deemed invalid, as illustrated in the following examples. With respect to a MB, the nodes can be categorized into five groups: (1) the target node, (2) parent nodes of the target, (3) child nodes of the target, (4) spousal nodes, which are parents of the children, and (5) other nodes, which are not part of the current MB. Incoming arcs into parents or spouses are not part of the MB structure and, hence operations that add such arcs are deemed invalid. Arcs between nodes not in the MB are not part of the MB structure and, hence operations that add such arcs are also deemed invalid. Figure 4 gives exhaustively the validity of the MB operators. Furthermore, the application of the delete arc or the reverse arc operators may lead to additional removal of arcs to produce a valid MB structure (see Figure 5 for an example).

As described in the previous section, the search for MB structures proceeds in two sequential phases. In phase 1 the candidate MB structures are scored with the Bayesian score (phase 1 score) shown in Equation 11. Since this phase selects the highest scoring MB structure at each iteration, it

X \ Y	T	P	C	S	O
T				✓	✓
P			✓		
C			✓*		
S	✓		✓		
O	✓		✓		

(a) Add arc X → Y

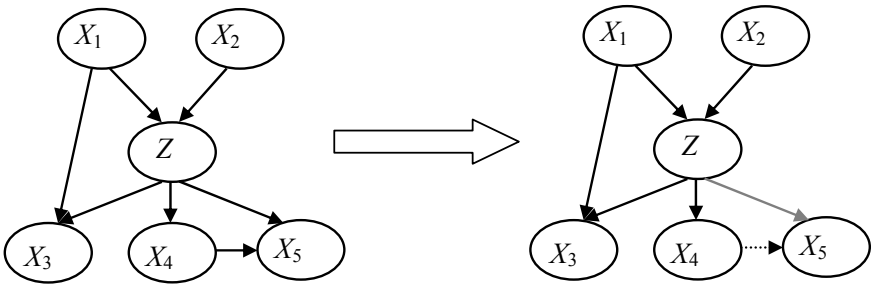
X \ Y	T	P	C	S	O
T			✓		
P	✓		✓		
C			✓		
S			✓		
O					

(b) Delete arc X → Y

X \ Y	T	P	C	S	O
T			✓		
P	✓				
C			✓*		
S					
O					

(c) Reverse arc X → Y

Figure 4: Constraints on the Markov blanket operators. The nodes are categorized into five groups: T = target, P = parent, C = child, S = spouse, and O = other (not in the Markov blanket of T). The cells with check marks indicate valid operations and are the only ones that need to be considered in generating candidate structures. The cells with an asterisk indicate that the operation is valid only if the resulting graph is acyclic.



(a) (b)

Figure 5: An example where the application of an operator leads to additional removal of arcs to produce a valid Markov blanket structure. Deletion of arc $Z \rightarrow X_5$ leads to removal of the arc $X_4 \rightarrow X_5$ since X_5 is no longer a part of the Markov blanket of Z. Reversal of the same arc also leads to removal of the arc $X_4 \rightarrow X_5$ since X_5 is now a parent and is precluded from having incoming arcs. Also, unless $X_4 \rightarrow X_5$ is removed there will be a cycle.

accumulates MB structures with high marginal likelihood. The purpose of this phase is to identify a set of MB structures that are highly probable, given data D.

Phase 2 searches for MB structures that change the current model-averaged estimate of $P(Z^t|\mathbf{x}^t, D)$ the most. The notion here is to find viable competing MB structures for making this posterior probability prediction. When no competitive MB structures can be found, the prediction is assumed to be stable. Phase 2 differs from the phase 1 in two aspects: it uses best-first search and it employs a different scoring function for evaluating candidate MB structures.

At the beginning of the phase 2, R contains MB structures that were generated in phase 1. Successors to the MB structures in R are generated, scored with the phase 2 score (described in detail below) and added to the priority queue Q. At each iteration of the search, the highest scoring MB structure in Q is removed from Q and added to R; all operations leading to legal MB structures are applied to it; the successor structures are scored with the phase 2 score; and the scored structures are added to Q. Phase 2 search terminates when no MB structure in Q has a score higher than some small value ε or when a period of time t has elapsed, where ε and t are user specified parameters.

In phase 2, the model score is computed as follows. Each successor MB structure G^* to be added to Q is scored based on how much it changes the current estimate of $P(Z^t|\mathbf{x}^t,D)$; this is obtained by model averaging over the MB structures in R. More change is better. Specifically, we use the Kullback-Leibler (KL) divergence between the two estimates of $P(Z^t|\mathbf{x}^t,D)$, one estimate computed with and another computed without G^* in the set of models over which the model averaging is carried out. The KL divergence, or relative entropy, is a quantity which measures the difference between two probability distributions (Cover and Joy, 2006). Thus, the phase 2 score for a candidate MB structure G^* is given by:

$$f(R,G^*) = \mathrm{KL}(p||q) \equiv \sum_x p(x) \log \frac{p(x)}{q(x)},$$

where

$$p(x) = \sum_{G \in R} P(Z^t|\mathbf{x}^t,G,D) \frac{P(G|D)}{\sum_{G' \in R} P(G'|D)}$$

and

$$q(x) = \sum_{G \in R \cup G^*} P(Z^t|\mathbf{x}^t,G,D) \frac{P(G|D)}{\sum_{G' \in R \cup G^*} P(G'|D)}.$$

Using Equation 11 the term $P(G|D)$ that appears in $p(x)$ and $q(x)$ can be substituted with the term $score(G;D)$. Using this substitution, the score for G^* is:

$$f(R,G^*) = \mathrm{KL}(p||q) \equiv \sum_x p(x) \log \frac{p(x)}{q(x)}, \tag{14}$$

where

$$p(x) = \sum_{G \in R} P(Z^t|\mathbf{x}^t,G,D) \frac{score(G;D)}{\sum_{G' \in R} score(G';D)}$$

and

$$q(x) = \sum_{G \in R \cup G^*} P(Z^t|\mathbf{x}^t,G,D) \frac{score(G;D)}{\sum_{G' \in R \cup G^*} score(G';D)}.$$

The pseudocode for the two-phase search procedure used by ISMB algorithm is given in Figure 6.

```
ProcedureSearchForISMB
    // phase 1: greedy hill-climbing search
    R ← empty set
    BestModel ← empty MB (graph containing only the target node)
    Score BestModel with phase 1 score
    BestScore ← phase 1 score of BestModel
    Add BestModel to R

    Do
        For every possible operator O that can be applied to BestModel
            Apply O to BestModel to derive Model
            Score Model with phase 1 score
            ModelScore ← phase 1 score of Model
            If ModelScore > BestScore
                BestModel ← Model
                BestScore ← ModelScore
                FoundBetterModel ← True
            End if
        End for
        If FoundBetterModel is True
            Add BestModel to R
        Else
            Terminate do
        End if
    End do

    // phase 2: best-first search
    Q ← empty priority queue with maximum capacity w
    Generate all successors for the MB structures in R and add them to Q
    Score all MB structures in Q with phase 2 score

    Do while elapsed time < t
        BestModel ← remove MB structure with highest phase 2 score from Q
        BestScore ← phase 2 score of BestModel
        For every possible operator O that can be applied to BestModel
            Apply O to BestModel to derive Model
            Score Model with phase 2 score
            Add Model to Q
        End for
        If BestScore > ε
            Add BestModel to R
        Else
            Terminate do
        End if
    End do

    Return R
```

Figure 6: Pseudocode for the two-phase search procedure used by the ISMB algorithm. Phase 1 uses greedy hill-climbing search while phase 2 uses best-first search.

5.8 Complexity of the ISMB Algorithm

For one instance, the ISMB algorithm runs in $O(bdmn)$ time and uses $O((w+d)mn)$ space, where m is the number of instances in the training data set D, n is the number of domain variables, d is the

total number of iterations of the search in the two phases 2, b (the branching factor) is the maximum number of successors generated from a MB structure, and w is the capacity of the priority queue Q.

5.8.1 TIME COMPLEXITY

At each iteration of the search, a maximum of b successor MB structures are generated. For d iterations of the search, the number of MB structures generated and scored with the phase 1 score is $O(bd)$. Note that both phases of the search require successor MB structures to be scored with the phase 1 score.

Since the phase 1 score decomposes over the MB nodes, to compute it for a newly generated MB structure only those MB nodes whose parent nodes have changed need be evaluated. The number of MB nodes that need to be evaluated is either one (when the *add* or *remove* operator is applied) or two (when the *reverse* operator is applied). Computing the phase 1 score for a MB node entails estimating the parameters for that node and calculating the marginal likelihood from those parameters. Estimating the parameters requires one pass over D and takes $O(mn)$ time which determines the time complexity of the phase 1 score.

The phase 2 score computes the effect of a candidate MB structure on the model averaged estimate of the distribution of the target variable. This requires doing inference for the target node in a MB that contains all measured variables which takes $O(n)$ since at most n nodes influence the target distribution and hence at most n sets of parameters need be retrieved. Computing both phase 1 and phase 2 scores for a MB structure therefore takes $O(mn)$ time. Thus, the total time required by the ISMB algorithm that runs for d iterations of the search and generates b MB structures at each iteration is $O(bdmn)$. However, the branching factor b is $O(n^2)$ and d is $O(n)$ and hence the overall complexity is $O(mn^4)$. This complexity limits that algorithm's applicability to data sets of small to medium dimensionality with up to several hundred variables.

5.8.2 SPACE COMPLEXITY

The ISMB algorithm searches in the space of MB structures using greedy hill-climbing search for phase 1 and best-first search with a priority queue of capacity w for phase 2. For d iterations of the search, the maximum number of MB structures that is stored is $O(w+d)$. The space required for each MB structure is linear in the number of its parameters.

For a given MB node, the number of parameters (using a conditional probability table) is exponential in the number of its parent nodes. However, the number of distinct parameters cannot be greater than the number of instances m in the training data D; the remaining parameters for a node have a single default value. Thus, the space required for the parameters of a MB node is $O(m)$. In a domain with n variables, a MB structure can have up to n nodes and thus requires space of $O(mn)$. In total, the space required by the ISMB algorithm that runs for d iterations of the search is $O((w+d)mn)$.

6. Evaluation of the ISMB Algorithm

This section describes the evaluation of the ISMB algorithm on a synthetic data set and several data sets from the UCI Machine Learning repository (UCI data sets). We first describe the preprocessing of variables, the evaluation measures and the comparison algorithms.

6.1 Preprocessing

Any instance that had one or more missing values was removed from the data set, as was done by Friedman et al. (1997). Sixteen of the 21 UCI data sets have no missing values and no instances were removed. In the remaining five data sets, removal of missing values resulted in a decrease in the size of the data set of less than 10%. After the removal of instances with missing values, the data sets were evaluated with two stratified applications of 10-fold cross-validation. Hence, each data set was split twice into 10 stratified training and test folds to create a total of 20 training and test folds. All experiments were carried out on the same set of 20 training and test folds. All target variables in all the data sets are discrete. However, some of the predictor variables are continuous. All continuous variables were discretized using the method described by Fayyad and Irani (1993). The discretization thresholds were determined only from the training sets and then applied to both the training and test sets.

6.2 Performance Measures

The performance of the ISMB algorithm was evaluated on two measures of discrimination (i.e., prediction under 0-1 loss) and three probability measures. The discrimination measures used are the misclassification error and the area under the ROC curve (AUC). For multiple classes, we used the method described by Hand and Till (2001) for computing the AUC. The discrimination measures evaluate how well an algorithm differentiates among the various classes (or values of the target variable). The probability measures considered are the logarithmic loss, squared error, and calibration. The closer the measure is to zero the better. For the multiclass case, we computed the logarithmic loss as described by Witten and Frank (2005) and the squared error as described by Yeung et al. (2005). For calibration, we used the CAL score that was developed by Caruana and Alexandru (2004) and is based on reliability diagrams. The probability measures indicate how well probability predictions correspond to reality. For example, consider a subset C of test instances in which target outcome is predicted to be positive with probability p. If a fraction p of C actually has a positive outcome, then such performance will contribute toward the probability measures being low. A brief description of the measures is given in Table 1.

Performance measure	Range	Best score
Misclassification error	$[0, 1]$	0
Area under the ROC curve (AUC)	$[0, 1]$	1
Logarithmic loss	$[0, \infty)$	0
Squared error	$[0, 1]$	0
Calibration score (CAL)	$[0, 1]$	0

Table 1: Brief description of the performance measures used in evaluation of the performance of the algorithms.

6.3 Comparison Algorithms

The performance of the instance-specific algorithms was compared to the following methods: nave Bayes (NB), C4.5 decision tree (DT), logistic regression (LR), neural networks (NN), k-Nearest Neighbor (kNN), Lazy Bayesian Rules (LBR), and AdaBoost (AB). The first four are representative

population-wide methods, the next two are examples of instance-specific methods, and AB is an ensemble method. *k*NN is a similarity-based method. The LBR algorithm induces a rule tailored to the features of the test instance that is then used to classify it, and is an example of a model-based instance-specific method that performs model selection. For all the seven comparison methods, we used the implementations in the Weka software package (version 3.4.3) (Witten and Frank, 2005). We used the default settings provided in Weka for NB, DT, and LR. For NN, we set the number of hidden nodes to $(n+c)/2$ where n is the number of predictor variables and c is the number of classes, the learning rate to 0.3 and the momentum to 0.2 (these are the default settings in Weka) and the number of iterations to 1000 since this setting resulted in slightly better performance than the default setting of 500. For *k*NN, we used the Weka setting that identifies the best value for k (i.e., the number of neighbors) by way of cross validation. For AB, we used Weka's AdaBoostM1 procedure with the decision tree J48 as the base classifier and the number of iterations set to $n/log(m)$, where n is the number of variables and m is the number of instances in the training data set. We did not perform variable selection as a pre-processing step before applying the above classification methods. However, DT, LBR and AB perform variable selection as part of the model learning procedure, while the other the methods do not.

Three versions of the ISMB algorithm were used in the experiments described later in this section, and they are listed in Table 2. The ISMB algorithm performs selective model averaging to estimate the distribution of the target variable of the instance at hand as described in Section 5. The ISMB-MS algorithm is a *model selection* version of the ISMB algorithm. It chooses the MB structure that has the highest posterior probability from those found by the ISMB algorithm in the two-phase search, and uses that single model to estimate the distribution of the target variable of the instance at hand. Comparing the ISMB algorithm to the ISMB-MS algorithm measures the effect of approximating selective model averaging by using model selection. When the training data set is large the performance of the ISMB algorithm and the ISMB-MS algorithm may be similar if a single model with a relatively large posterior probability overwhelms the contributions of the remaining models during model averaging.

Acronym	Algorithm	Phase 1	Phase 2	Prediction
ISMB	Instance-specific Markov blanket	Is non-instance-specific Uses greedy hill-climbing search Uses phase 1 score	Is instance-specific Uses best-first search Uses phase 2 score	By model averaging over models selected in phase 1 and phase 2
ISMB-MS	Instance-specific Markov blanket - Model Selection	Same as ISMB	Same as ISMB	Based on the highest scoring model from models found by ISMB
NISMB	Non-instance-specific Markov blanket	Same as ISMB	Is non-instance-specific Uses best-first search Uses phase 1 score	By model averaging; number of selected models is the same as in ISMB

Table 2: Three versions of the ISMB algorithm.

The NISMB algorithm is the *non-instance-specific* (i.e., population-wide) version of the ISMB algorithm. Phase 1 of the NISMB algorithm is identical to that of the ISMB algorithm. In phase 2, the NISMB algorithm accumulates the same number of MB models as the ISMB algorithm except that the models are identified on the basis of the non-instance-specific phase 1 score. Thus, the NISMB algorithm averages over the same number of models as the ISMB algorithm. Comparing the ISMB algorithm to the NISMB algorithm measures the effect of the instance-specific heuristic on the performance of model averaging.

6.4 Evaluation on a Synthetic Data Set

This section describes the evaluation of the ISMB algorithm on a small synthetic data set. The synthetic domain consists of five binary variables A, B, C, D, Z where Z is a deterministic function of the other variables:

$$Z = A \vee (B \wedge C \wedge D).$$

On such a small data set it is possible to perform model averaging over all models, and this establishes the best possible prediction performance that is attainable using MB models. The training and the test sets used in the experiments are shown in Figure 7. The training set simulates a low occurrence of $A = \mathrm{T}$ (only five out of 69 instances have $A = \mathrm{T}$), and the test set consists of three instances of $A = \mathrm{T}$ which are not present in the training set.

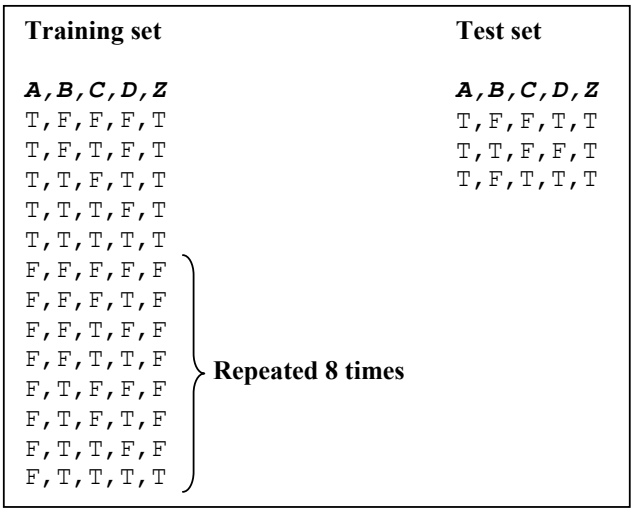


Figure 7: Training and test data sets derived from the deterministic function $Z = A \vee (B \wedge C \wedge D)$. The training set contains a total of 69 instances and the test set a total of three instances as shown; the test instances are not present in the training set. The training set simulates low prevalence of $A = \mathrm{T}$ since only five of the 69 instances have this variable-value combination.

The following algorithms were used in the experiments: (1) a complete model averaged version of the ISMB algorithm where model averaging is carried out over all 3567 possible MB structures, (2) the ISMB algorithm, (3) the ISMB-MS algorithm, and (4) the NISMB algorithm.

The settings used for the ISMB algorithm are as follows:

- Phase 1: As described in Section 5.
- Phase 2: The model score for phase 2 is computed using Equation 14 that is based on KL-divergence. Phase 2 uses best-first search with a priority queue Q whose maximum capacity w was set to 1000. Phase 2 search terminates when no MB structure in Q has a phase 2 score higher than $\varepsilon = 0.001$ for 10 consecutive iterations of the search. The maximum period of running time t for phase 2 was not specified since the algorithm terminated in a reasonable period of time with the specified value for ε.
- The predicted distribution for the target variable Z of the test instance is computed using Equation 13; for each MB structure the parameters are estimated using Equation 6.

The results are given in Table 3. All performance measures except the AUC were computed for the test set of three instances. The AUC could not be computed since all the instances in the test set are from the same class, $Z = \mathrm{T}$. The results from complete model averaging represent the best achievable expected performance that could be achieved by the ISMB algorithm. The ISMB and the NISMB algorithms that average over a subset of all models had poorer performance than complete model averaging but performed better than ISMB-MS. However, the ISMB algorithm improved over the performance of the NISMB algorithm. Though both methods average over the same number of models, the ISMB algorithm uses the instance-specific phase 2 score to choose phase 2 models while the ISMB algorithm uses the non-instance-specific phase 1 score to choose both phase 1 and phase 2 models. The phase 2 models chosen by the ISMB algorithm are potentially different for each test instance in contrast to the NISMB algorithm which selects the same models irrespective of the test instance. These results, while limited in scope, provide support that the instance-specific search for models may be able to choose models that better approximate the distribution of the target variable of the instance at hand.

Performance measure	ISMB complete model averaged	ISMB	ISMB-MS	NISMB
Misclassification error	0.0000	0.0000	0.3333	0.3333
AUC	-	-	-	-
Logarithmic loss	0.0406	0.0505	0.0596	0.0585
Squared error	0.0684	0.0783	0.0902	0.0862
CAL score	0.3720	0.4092	0.4534	0.4284

Table 3: Results obtained from the training and test sets that are given in Figure 7. The AUC could not be computed since the test set instances are all from a single class. Results in the first column are obtained by model averaging over all 3567 MBs.

Figure 8 plots the estimate of $P(Z^t = \mathrm{T}|\mathbf{x}^t, D)$ for each test instance t as it varies with each addition of a model to the set of models being averaged over. A second curve plots the model score as the logarithmic posterior probability of the model given the data; this score measures the relative contribution of the model to the final estimate of $P(Z^t = \mathrm{T}|\mathbf{x}^t, D)$. Each row in the figure contains a pair of plots for a single test instance, the plot on the left is obtained from the ISMB algorithm and

the corresponding plot on the right is obtained from the NISMB algorithm. The plot for the estimate of $P(Z^t = \mathrm{T}|\mathbf{x}^t, D)$ is shown in black while the plot for the model score is shown in gray. In each plot, on going from left to right, the estimate of $P(Z^t = \mathrm{T}|\mathbf{x}^t, D)$ initially fluctuates considerably and then settles to a stable estimate as the number of models providing the estimate increases. In the first two test instances the final estimates of $P(Z^t = \mathrm{T}|\mathbf{x}^t, D)$ obtained from the instance-specific and non-instance-specific model averaging respectively are very close; both the ISMB and the NISMB algorithms predicted the value of Z correctly as T. In the third test instance, the final estimates of $P(Z^t = \mathrm{T}|\mathbf{x}^t, D)$ are quite different; the ISMB algorithm predicted the value of Z correctly as T while the NISMB algorithm predicted the value of Z incorrectly as F.

6.5 Evaluation on UCI Data Sets

We now describe the evaluation of the ISMB algorithm on 21 data sets from the UCI Machine Learning repository (UCI data sets) (Frank and Asuncion, 2010). The selected UCI data sets have between four and 60 predictor variables and a single target variable that has between two and seven classes. The size of the data sets, the number and type of predictor variables, and the number of classes (states) taken by the target variable are given in Table 4 The performance of the ISMB algorithm is compared to that of the ISMB-MS and the NISMB algorithms, and also to that of the seven comparison machine learning methods described in Section 6.3.

6.5.1 EXPERIMENTAL DESIGN

The experimental design is as follows:

- For each data set, a total of 10 machine learning algorithms were run: ISMB, ISMB-MS, NISMB, NB, DT, LR, NN, *k*NN, LBR and AB.
- The data sets used in the experiments are the 21 UCI data sets listed in Table 4.
- Summary statistics were measured using 10-fold stratified cross-validation done twice for a total of 20 training-test pairs. The summary statistics were computed for misclassification error, the AUC, logarithmic loss, squared error and the CAL score.
- The statistical tests performed were (1) significance testing with the Wilcoxon paired-samples signed ranks test, and (2) effect size testing with paired-samples *t* test.

The settings for the ISMB algorithm are the same as those stated in Section 6.4 for the synthetic data evaluation.

6.5.2 RESULTS

Table 5 gives the average number of models selected by the ISMB and the NISMB algorithms in each of the phases for each data set. The average number of models varies from 17.99 for the iris data set (with four predictor variables) to 89.38 for the lymphography data set (with 18 predictor variables).

Tables 6 to 10 report the means of the misclassification error, the AUC, logarithmic loss, squared error and the CAL score respectively for the ISMB algorithm, its variants and the comparison algorithms. In each table, a row represents a data set and a column represents an algorithm. The last

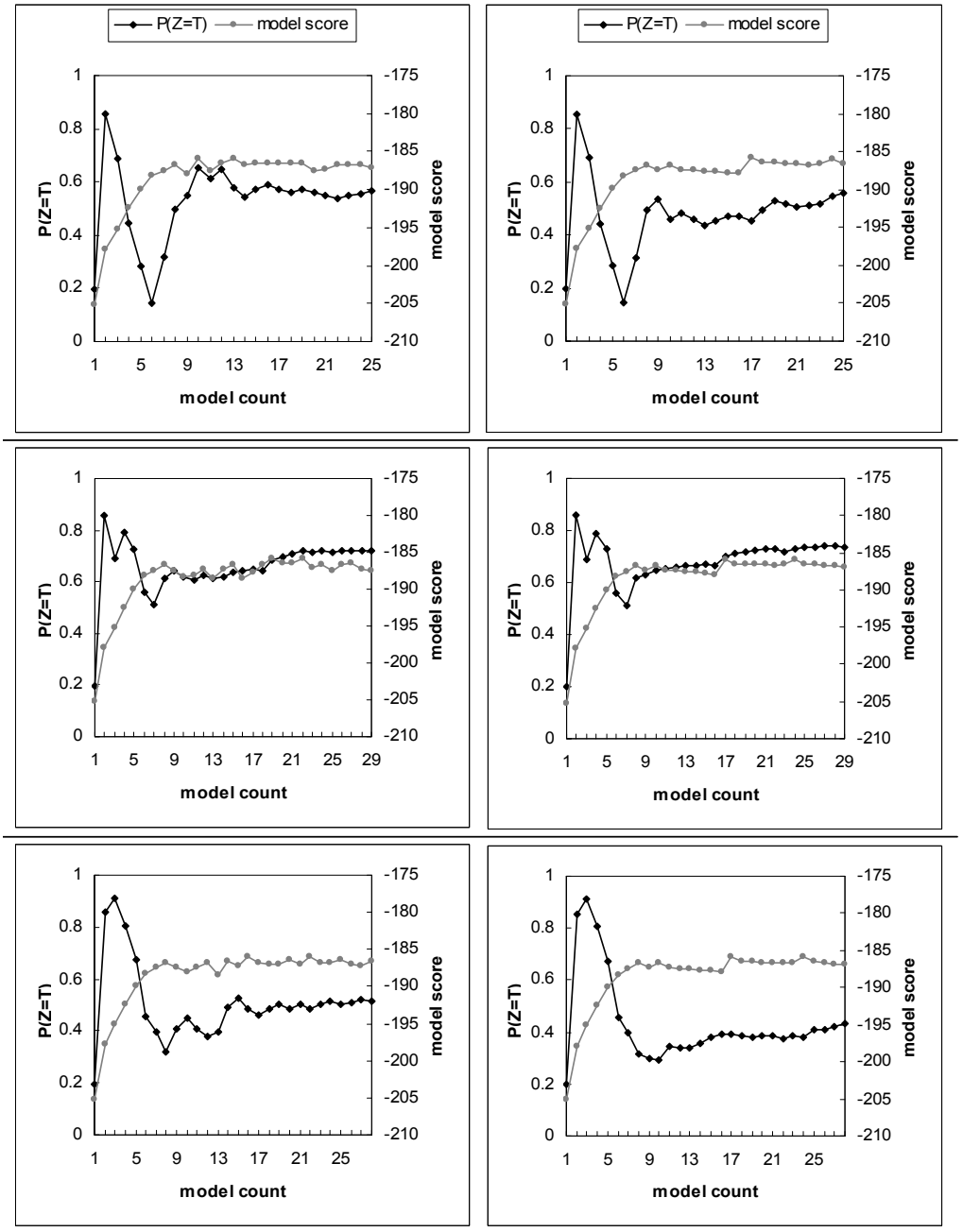


Figure 8: Plots of model averaged estimate of $P(Z^t = \mathrm{T}|\mathbf{x}^t, D)$ that is abbreviated as $P(Z = \mathrm{T})$ and model score obtained by ISMB and NISMB algorithms on the three test cases given in Figure 7. Each row represents a single test case with the plot on the left obtained from the ISMB algorithm and the plot on the right obtained from the NISMB algorithm. The value of the final averaged estimate of $P(Z^t = \mathrm{T}|\mathbf{x}^t, D)$ is the point where the darker curve meets the Y-axis on the right.

Data Set	# Predictors (cnt + dsc = total)	# Classes	# Cases
australian	6 + 8 = 14	2	690
breast-cancer	9 + 0 = 9	2	683
cleveland	6 + 9 = 13	2	296
corral	0 + 6 = 6	2	128
crx	6 + 9 = 15	2	653
diabetes	8 + 0 = 8	2	768
flare	0 + 10 = 10	2	1066
german	7 + 13 = 20	2	1000
glass2	9 + 0 = 9	2	163
glass	9 + 0 = 9	7	214
heart	13 + 0 = 13	2	270
hepatitis	6 + 13 = 19	2	80
iris	4 + 0 = 4	3	150
lymphography	0 + 18 = 18	4	148
pima	8 + 0 = 8	2	768
postoperative	1 + 7 = 8	3	87
sonar	60 + 0 = 60	2	208
vehicle	18 + 0 = 18	4	846
vote	0 + 16 = 16	2	435
wine	13 + 0 = 13	3	178
zoo	0 + 16 = 16	7	101

Table 4: Description of the 21 UCI data sets used in the experiments. In the column on predictors, the number of continuous (cnt) and discrete (dsc) predictors as well as the total number of predictor variables (excluding the target variable) are given. In the column on instances, the numbers of instances used in the experiments are given; this may be less than the total number of instances in the original UCI data set since instances with missing values were removed.

row in each table gives for each algorithm the overall mean of the specified performance measure across all the data sets. From the tables, it is seen that on all five performance measures, the ISMB algorithm achieved a better overall average score than each of the other algorithms.

Tables 11 and 12 report results from pair-wise comparisons of the performance of the algorithms on all the data sets that are aimed at assessing the statistical significance and the magnitude of the observed differences in the measures. Table 11 reports results from a two-sided Wilcoxon paired-samples signed ranks test, and Table 12 reports results from a two-sided paired-samples t test.

Table 13 reports the running times of the ISMB and the comparison algorithms. The experiments were performed on a server with 8 GB of RAM and two dual core Pentium processors of 3 GHz each that were running the Windows XP operating system. The algorithms were restricted to a single core in all the experiments. Averaged over all the data sets, the ISMB took approximately 2 minutes for a test instance.

We ran additional experiments on the first seven data sets (see Table 4) to analyze the sensitivity of the ISMB algorithm to the parameters w (queue capacity) and ε (change in Phase 2 score). For w, we evaluated values of 100, 200, 400, 800, 1600, 3200 and 6400. The performance on all the evaluation measures peaked at values of 800 or 1600 and beyond 1600 no further improvement was

Data Set	# models phase 1	# models phase 2	# models phases 1 and 2
australian	28.55	11.00	39.55
breast-cancer	18.85	10.15	29.00
cleveland	20.45	11.99	32.44
corral	10.65	15.03	25.68
crx	32.10	13.42	45.52
diabetes	11.65	10.03	21.68
flare	20.75	11.44	32.19
german	22.45	19.23	41.68
glass2	12.05	13.26	25.31
glass	15.80	10.73	26.53
heart	18.50	11.32	29.82
hepatitis	27.45	26.63	54.08
iris	7.25	10.74	17.99
lymphography	51.55	37.83	89.38
pima	40.40	16.97	57.37
postoperative	12.00	10.02	22.02
sonar	11.65	10.09	21.74
vehicle	1.15	21.09	22.24
vote	59.80	18.44	78.24
wine	39.30	10.73	50.03
zoo	45.55	13.53	59.08

Table 5: Average number of models in phases 1 and 2 over which averaging is carried out by the ISMB and NISMB algorithms. Both algorithms average over the same number of models in each phase. Both algorithms select the same models in phase 1 but potentially different models in phase 2. The number of models in phases 1 and 2 is the sum of the models selected in the two phases.

seen. For ε, we evaluated values of 1.0, 0.1, 0.01, 0.001, 0.0001 and 0.00001. The performance improved as ε decreased until 0.001 or 0.0001, but did not improve further for smaller values of ε.

The results are encouraging in that they show that the ISMB algorithm never underperformed on any performance measure when compared to the other learning methods including the variants of the ISMB algorithm that do model selection and non-instance-specific model averaging. For misclassification error, logarithmic loss, squared error and the CAL score, the mean difference is always negative which denotes that the ISMB algorithm always has a lower score on these measures. For the AUC, the difference is always positive which means that the ISMB algorithm always has a higher AUC. However, all mean differences are not statistically significant at the 0.05 level as can be seen by the p-values in Tables 11 and 12. The best performance is seen in logarithmic loss where the ISMB algorithm significantly outperforms all other methods, followed by squared error and CAL score where the ISMB algorithm significantly outperforms many of the methods. On misclassification error and the AUC, the ISMB algorithm has smaller performance gains.

Overall, the ISMB algorithm significantly improved on the probabilities of the predictions while maintaining or slightly improving on discrimination over all other algorithms used in the experiments. The non-instance-specific NISMB algorithm had inferior performance on logarithmic loss and squared error but similar performance on the other measures when compared to the ISMB algorithm. Both the ISMB and the NISMB algorithms average over the same number of models and

Data Set	ISMB	ISMB-MS	NISMB	NB	DT	LR	NN	*k*NN	LBR	AB
australian	0.1457	0.1457	0.1435	0.1449	0.1333	0.1486	0.1848	0.1457	0.1471	0.1453
breast-cancer	0.0256	0.0271	0.0256	0.0256	0.0403	0.0337	0.0373	0.0286	0.0256	0.0264
cleveland	0.1740	0.1791	0.1740	0.1655	0.2095	0.1655	0.1993	0.1791	0.1655	0.1985
corral	0.0000	0.0156	0.0000	0.1328	0.0508	0.1289	0.0000	0.0977	0.1250	0.0000
crx	0.1547	0.1577	0.1485	0.1348	0.1317	0.1424	0.1692	0.1485	0.1340	0.1308
diabetes	0.2116	0.2129	0.2142	0.2201	0.2194	0.2135	0.2272	0.2201	0.2207	0.2244
flare	0.1806	0.1834	0.1825	0.2012	0.1735	0.1721	0.2054	0.1806	0.1750	0.1730
german	0.2580	0.2585	0.2580	0.2445	0.2845	0.2425	0.2980	0.2695	0.2475	0.2818
glass2	0.1503	0.1564	0.1472	0.1595	0.1933	0.1442	0.1442	0.1411	0.1503	0.1503
glass	0.2150	0.2220	0.2196	0.2687	0.2500	0.2547	0.2220	0.2173	0.2500	0.2420
heart	0.1778	0.1778	0.1778	0.1630	0.1870	0.1630	0.1963	0.1741	0.1630	0.1724
hepatitis	0.0938	0.1000	0.1000	0.1375	0.1250	0.1375	0.1688	0.0688	0.1375	0.1040
iris	0.0567	0.0600	0.0633	0.0533	0.0600	0.0567	0.0633	0.0633	0.0533	0.0600
lymphography	0.1622	0.1486	0.1622	0.1486	0.2365	0.2365	0.1622	0.1622	0.1520	0.1622
pima	0.2155	0.2135	0.2142	0.2214	0.2259	0.2148	0.2389	0.2246	0.2227	0.2224
postoperative	0.3391	0.3851	0.3391	0.3103	0.2989	0.3736	0.4138	0.3333	0.3103	0.2111
sonar	0.1635	0.1659	0.1731	0.1490	0.1659	0.1442	0.1611	0.1707	0.1490	0.1742
vehicle	0.2600	0.2577	0.2612	0.3712	0.2843	0.2914	0.2825	0.2766	0.2784	0.2923
vote	0.0453	0.0582	0.0453	0.0927	0.0388	0.0733	0.0711	0.0819	0.0927	0.0438
wine	0.0084	0.0084	0.0056	0.0112	0.0702	0.0253	0.0169	0.0281	0.0112	0.0617
zoo	0.0347	0.0396	0.0347	0.0644	0.0792	0.0594	0.0495	0.0347	0.0644	0.0658
average	0.1463	0.1511	0.1471	0.1629	0.1647	0.1629	0.1672	0.1546	0.1560	0.1496

Table 6: Mean misclassification errors of different algorithms based on 10-fold cross-validation done twice. The bottom row gives the average misclassification errors. Best results are underlined.

both select the same models in phase 1 of the search. In phase 2 of the search, while the number of selected models is the same, the two methods identify potentially different models. This provides evidence that the models selected in phase 2 by the ISMB algorithm, using instance-specific search, are able to improve the performance of the ISMB algorithm over the already good performance obtained by the NISMB algorithm. Of note, LBR, which is an instance-specific approach that performs model selection, is tied with ISMB on mean error and comes second after ISMB on AUC, but it performs more poorly on the probabilistic measures.

7. Discussion

This paper described the development and evaluation of a new approach for learning predictive models that are relevant to a single instance. The instance-specific method we developed uses MB models, carries out selective BMA to predict the outcome of interest for the instance at hand, and employs an instance-specific heuristic to locate a set of suitable models to average over. The essence of the instance-specific method lies in the model score used in phase 2 of the search. This score is sensitive to both the posterior probability of the model and the predicted distribution for the outcome variable of the instance at hand. Typically, methods that evaluate models with a score employ a score that is sensitive only to the fit of the model to the training data and not to the prediction of the outcome variable.

Data Set	ISMB	ISMB-MS	NISMB	NB	DT	LR	NN	*k*NN	LBR	AB
australian	0.9315	0.9303	0.9313	0.9200	0.9032	0.9187	0.8937	0.9092	0.9186	0.9172
breast-cancer	0.9926	0.9922	0.9925	0.9933	0.9613	0.9879	0.9818	0.9930	0.9933	0.9802
cleveland	0.9098	0.9079	0.9084	0.9141	0.7952	0.9089	0.8781	0.8995	0.9141	0.8350
corral	1.0000	0.9997	1.0000	0.9252	0.9916	0.9459	1.0000	0.9827	0.9373	0.9932
crx	0.9303	0.9280	0.9302	0.9301	0.9087	0.9138	0.9002	0.9057	0.9302	0.9140
diabetes	0.8468	0.8468	0.8466	0.8438	0.7991	0.8439	0.8311	0.8148	0.8423	0.8004
flare	0.7289	0.7288	0.7261	0.7557	0.4916	0.7451	0.6445	0.6797	0.7520	0.7034
german	0.7662	0.7633	0.7641	0.7903	0.6736	0.7839	0.7340	0.7442	0.7891	0.7911
glass2	0.8703	0.8653	0.8700	0.8769	0.7982	0.8845	0.8483	0.8384	0.8826	0.8744
glass	0.9364	0.9361	0.9361	0.9408	0.8834	0.9101	0.9241	0.9112	0.9434	0.9488
heart	0.9055	0.9049	0.9073	0.9106	0.8239	0.9032	0.8649	0.8791	0.9106	0.8332
hepatitis	0.9225	0.9262	0.9237	0.9013	0.8203	0.7784	0.8436	0.8792	0.8970	0.9004
iris	0.9890	0.9900	0.9905	0.9938	0.9629	0.9846	0.9785	0.9886	0.9938	0.9808
lymphography	0.9139	0.9156	0.9173	0.9193	0.7741	0.8571	0.9192	0.9087	0.9175	0.8830
pima	0.8431	0.8424	0.8424	0.8450	0.7977	0.8456	0.8237	0.8134	0.8449	0.8284
postoperative	0.5026	0.4943	0.4538	0.5035	0.4228	0.4515	0.4113	0.3665	0.5035	0.4975
sonar	0.9203	0.9204	0.9217	0.9343	0.8521	0.9275	0.9331	0.9132	0.9345	0.9142
vehicle	0.9234	0.9228	0.9235	0.8655	0.8761	0.9016	0.8931	0.9032	0.9109	0.8965
vote	0.9875	0.9850	0.9854	0.9684	0.9578	0.9582	0.9871	0.9735	0.9660	0.9498
wine	0.9994	0.9994	0.9994	1.0000	0.9660	0.9967	0.9994	0.9981	1.0000	1.0000
zoo	0.9994	0.9992	0.9992	0.9989	0.9565	0.9967	0.9916	0.9995	0.9989	0.9622
average	0.8962	0.8952	0.8938	0.8919	0.8293	0.8783	0.8705	0.8715	0.8943	0.8764

Table 7: Mean AUCs of different algorithms based on 10-fold cross-validation done twice. The bottom row gives the average AUCs. Best results are underlined.

The experimental results demonstrate that the ISMB algorithm improves prediction of the target variable on a variety of performance measures when compared to several population-wide predictive algorithms. The greatest improvements occur in logarithmic loss and squared error, followed by good improvement in calibration and smaller improvements in misclassification error and the AUC. BMA had better performance than Bayesian model selection, and within model averaging, instance-specific BMA had better performance than non-instance-specific BMA though the improvement is not as large as that of model averaging over model selection. The improved performance by ISMB may arise from not only the model averaging but also from the variable selection that is performed implicitly by the Markov blanket models. Both these components likely explain the better performance of ISMB over comparison methods such as NB, LR and *k*NN that do not perform variable selection. However, the superiority of ISMB over ISMB-MS suggests that model averaging is an important component in the improved performance of the former. We have also evaluated ISMB on several medical data sets and obtained good results (Visweswaran et al., 2010).

Several situations are possible where the instance-specific method has no advantage over a population-wide method. As one example, in a domain where complete BMA is tractable and model averaging is carried out over all models in the model space, a search heuristic that selects a subset of models such as the one used by the instance-specific method is superfluous. Typically, in real life domains, complete BMA over all models is not tractable due to the enormous number of models in the model space. Thus, the ISMB algorithm is useful for selective model averaging where it identifies a potentially relevant set of models that is predictive of the instance at hand. As another

Data Set	ISMB	ISMB-MS	NISMB	NB	DT	LR	NN	*k*NN	LBR	AB
australian	0.3390	0.3456	0.3417	0.4476	0.4091	0.7136	0.4263	0.8627	0.4482	0.3850
breast-cancer	0.1068	0.1138	0.1083	0.2497	0.2955	0.1485	0.1205	0.2138	0.2497	0.2772
cleveland	0.3925	0.4067	0.4021	0.4491	1.3001	0.6500	0.4584	0.8625	0.4491	0.7044
corral	0.1018	0.1101	0.0989	0.3326	0.1475	0.2753	0.1542	0.0175	0.3130	0.1280
crx	0.3451	0.3564	0.3525	0.4113	0.3783	0.9377	0.4678	0.8747	0.4018	0.3845
diabetes	0.4601	0.4606	0.4604	0.4809	0.5497	0.4588	0.6039	0.5028	0.4826	0.4478
flare	0.4282	0.4294	0.4314	0.5904	0.4879	0.4042	0.5333	0.5858	0.5182	0.4032
german	0.5331	0.5413	0.5377	0.5213	1.4604	0.5229	0.5801	1.5415	0.5221	0.7981
glass2	0.4238	0.4302	0.4246	0.4532	0.8498	0.4154	0.8853	0.4562	0.4447	0.4530
glass	0.7112	0.7239	0.7113	0.7697	2.3005	4.0749	1.3612	0.8685	0.7264	0.9452
heart	0.3996	0.4069	0.3973	0.4560	0.6920	0.3907	0.6109	0.8483	0.4560	0.3788
hepatitis	0.2396	0.2517	0.2583	0.4247	0.6122	17.7871	0.3562	0.6253	0.4272	0.2548
iris	0.1560	0.1909	0.1620	0.1621	0.5287	0.7579	0.5770	0.2240	0.1621	0.1712
lymphography	0.4100	0.4289	0.4430	0.4282	2.9112	21.6371	0.5765	0.7272	0.4409	0.7422
pima	0.4647	0.4657	0.4657	0.4793	0.5268	0.4572	0.5873	0.5114	0.4774	0.4880
postoperative	0.7381	0.7776	0.7287	0.7953	1.1395	2.8236	1.3339	1.9418	0.7953	0.9453
sonar	0.3573	0.3726	0.3743	0.4573	1.2814	0.5762	0.4170	0.5728	0.4554	0.4344
vehicle	0.5863	0.5900	0.5866	1.8645	2.3842	3.9997	1.0134	1.2590	0.7815	1.0042
vote	0.1393	0.1635	0.1588	0.6804	0.3028	5.5427	0.3171	0.2782	0.5629	0.1562
wine	0.0418	0.0402	0.0367	0.0303	0.8270	0.9593	0.1032	0.0409	0.0303	0.0531
zoo	0.1297	0.1202	0.1268	0.1474	1.1102	0.5325	0.0596	0.1595	0.1474	0.1130
average	0.3573	0.3679	0.3622	0.5063	0.9759	3.0507	0.5497	0.6654	0.4425	0.4604

Table 8: Mean logarithmic losses of different algorithms based on 10-fold cross-validation done twice. The bottom row gives the average logarithmic losses. Best results are underlined.

example, in a domain where features that are relevant are commonly present, selection of relevant variables may not be a problem. In such a situation, the variables selected by a population-wide method are likely to be relevant for predicting any future instance and the instance-specific method that performs model selection will likely select the same set of variables for each new instance.

Improvements in the phase 1 search may make the phase 2 search relatively less contributory to the overall performance. We believe that the greedy hill climbing approach used in phase 1 of ISMB serves as a useful starting point for investigating this algorithm. Nonetheless, such an approach may become trapped in local maxima, leading it to miss finding highly probable MB structures. To explore this issue a number of search strategies that augment local greedy search that have been successfully applied to learning BN structures can be tried, such as best-first search (Neapolitan, 2003), simulated annealing (Heckerman et al., 1995), tabu lists (Friedman et al., 1999), random restarts to escape the numerous local optima (Heckerman et al., 1995), and optimal reinsertion (Moore and Wong, 2003). Algorithms that have been developed specifically for learning MBs such as the Markov Blanket Bayesian Classifier (MBBC) (Madden, 2002a), HITON (Aliferis et al., 2003), the Incremental Association Markov Blanket (IAMB) (Tsamardinos and Aliferis, 2003), and the Min-Max Markov Blanket algorithm (MMMB) (Tsamardinos et al., 2006) are additional candidates for consideration. Investigating the use of such alternative search methods in phase 1 is an interesting open problem.

There are several open questions regarding the behavior of the instance-specific method. Characterizing theoretically the bias of the selective model averaged prediction of the instance-specific

Data Set	ISMB	ISMB-MS	NISMB	NB	DT	LR	NN	*k*NN	LBR	AB
australian	0.2054	0.2082	0.2060	0.2234	0.2066	0.2116	0.3062	0.2287	0.2287	0.2075
breast-cancer	0.0440	0.0449	0.0441	0.0474	0.0731	0.0542	0.0689	0.0484	0.0474	0.0622
cleveland	0.2433	0.2499	0.2462	0.2553	0.3516	0.2339	0.3364	0.2526	0.2553	0.3314
corral	0.0352	0.0463	0.0354	0.2056	0.0887	0.1836	0.0038	0.1051	0.1951	0.0787
crx	0.2081	0.2146	0.2087	0.2092	0.1965	0.2121	0.2948	0.2363	0.2078	0.1987
diabetes	0.2978	0.2981	0.2979	0.3073	0.3219	0.2978	0.3156	0.3315	0.3086	0.2978
flare	0.2619	0.2626	0.2652	0.3145	0.2846	0.2513	0.3203	0.2843	0.2700	0.2498
german	0.3526	0.3570	0.3555	0.3419	0.4196	0.3368	0.5104	0.3591	0.3433	0.4050
glass2	0.2469	0.2513	0.2468	0.2450	0.3116	0.2409	0.2572	0.2603	0.2393	0.2572
glass	0.3609	0.3635	0.3605	0.3823	0.4186	0.4363	0.4075	0.3880	0.3673	0.3857
heart	0.2444	0.2486	0.2420	0.2570	0.3113	0.2394	0.3273	0.2611	0.2570	0.2565
hepatitis	0.1410	0.1495	0.1534	0.2079	0.2170	0.2750	0.2579	0.1481	0.2090	0.1483
iris	0.0727	0.0828	0.0753	0.0751	0.1122	0.0942	0.1032	0.1086	0.0751	0.0834
lymphography	0.2391	0.2353	0.2433	0.2344	0.4162	0.4545	0.2687	0.2650	0.2406	0.2388
pima	0.3009	0.3011	0.3011	0.3065	0.3264	0.2968	0.3248	0.3332	0.3060	0.3130
postoperative	0.4772	0.5044	0.4748	0.4894	0.4525	0.6011	0.7221	0.6168	0.4894	0.4512
sonar	0.2349	0.2391	0.2369	0.2411	0.2887	0.2228	0.2764	0.2402	0.2405	0.2614
vehicle	0.3471	0.3481	0.3470	0.5805	0.4171	0.4109	0.4672	0.3934	0.4059	0.3815
vote	0.0788	0.0903	0.0810	0.1681	0.0703	0.1461	0.1172	0.1293	0.1529	0.0911
wine	0.0183	0.0158	0.0142	0.0191	0.1268	0.0503	0.0213	0.0407	0.0191	0.0255
zoo	0.0612	0.0652	0.0630	0.0860	0.1415	0.0991	0.0568	0.0406	0.0860	0.0877
average	0.2129	0.2179	0.2142	0.2475	0.2644	0.2547	0.2745	0.2415	0.2354	0.2292

Table 9: Mean squared errors of different algorithms based on 10-fold cross-validation done twice. The bottom row gives the average squared errors. Best results are underlined.

method is an open problem. In contrast, the bias of selective BMA over models that are chosen randomly is low. However, the variance of selective BMA over models that are chosen randomly is likely to be much larger than the variance of selective BMA over models chosen by the instance-specific method which is constrained to prefer models that are good fit to the training data. The results here support that as a practical matter ISMB is attaining a good balance between bias and variance.

The experimental work presented in this paper is a first step in exploring the utility of the instance-specific framework, and several directions of future work are possible. The computation of the phase 2 score (see Equation 14) requires a dissimilarity metric to compare the predictive distributions of the target variable in candidate MB structures. The current implementation of the ISMB algorithm uses KL divergence as the dissimilarity metric. The experimental results indicate that KL divergence optimizes most logarithmic loss and the largest improvement in performance is observed on this measure. Alternative dissimilarity metrics that may optimize other performance measures are worth exploring.

Data Set	ISMB	ISMB-MS	NISMB	NB	DT	LR	NN	*k*NN	LBR	AB
australian	0.0470	0.0459	0.0454	0.0775	0.0463	0.0440	0.0526	0.1423	0.0817	0.0454
breast-cancer	0.0146	0.0146	0.0144	0.0200	0.0261	0.0155	0.0114	0.0299	0.0200	0.0210
cleveland	0.0497	0.0630	0.0569	0.0930	0.0690	0.0295	0.0432	0.1543	0.0930	0.0632
corral	0.0583	0.0656	0.0561	0.0470	0.0505	0.0473	0.0162	0.0115	0.0368	0.0516
crx	0.0452	0.0518	0.0503	0.0711	0.0440	0.0394	0.0722	0.1354	0.0689	0.0430
diabetes	0.0403	0.0401	0.0411	0.0618	0.0633	0.0433	0.0813	0.0662	0.0590	0.0400
flare	0.0551	0.0546	0.0562	0.1260	0.0467	0.0414	0.0762	0.1000	0.0707	0.0404
german	0.0684	0.0696	0.0699	0.0625	0.1038	0.0504	0.0547	0.2363	0.0645	0.0942
glass2	0.0359	0.0395	0.0373	0.0644	0.0386	0.0322	0.0482	0.0561	0.0569	0.0349
glass	0.0188	0.0189	0.0186	0.0282	0.0223	0.0262	0.0258	0.0246	0.0241	0.0232
heart	0.0498	0.0585	0.0513	0.0913	0.0641	0.0321	0.0624	0.1385	0.0913	0.0524
hepatitis	0.0422	0.0294	0.0381	0.0488	0.0306	0.0462	0.0197	0.0492	0.0466	0.0288
iris	0.0110	0.0115	0.0114	0.0132	0.0188	0.0142	0.0219	0.0205	0.0132	0.0144
lymphography	0.0226	0.0259	0.0256	0.0326	0.0279	0.0863	0.0272	0.0512	0.0359	0.0269
pima	0.0532	0.0539	0.0539	0.0596	0.0660	0.0444	0.0960	0.0805	0.0586	0.0588
postoperative	0.0404	0.0358	0.0438	0.0436	0.0450	0.0707	0.0844	0.1175	0.0436	0.0430
sonar	0.0437	0.0656	0.0643	0.1042	0.0591	0.0814	0.0503	0.1336	0.1045	0.0535
vehicle	0.0479	0.0481	0.0480	0.1272	0.0654	0.0632	0.0567	0.0984	0.0690	0.0543
vote	0.0247	0.0285	0.0306	0.0722	0.0227	0.0520	0.0603	0.0346	0.0658	0.0235
wine	0.0062	0.0043	0.0054	0.0083	0.0247	0.0154	0.0256	0.0133	0.0083	0.0103
zoo	0.0065	0.0067	0.0069	0.0078	0.0094	0.0055	0.0029	0.0075	0.0078	0.0067
average	0.0372	0.0396	0.0393	0.0600	0.0450	0.0419	0.0471	0.0810	0.0533	0.0395

Table 10: Mean CAL scores of different algorithms based on 10-fold cross-validation done twice. The bottom row gives the average CAL scores. Best results are underlined.

Performance measure	ISMB-MS	NISMB	NB	DT	LR	NN	*k*NN	LBR	AB
Misclassification	-2.338	-0.776	-2.121	-2.070	-1.181	-3.861	-0.825	-0.368	-1.720
error	0.019	0.438	0.034	0.038	0.238	<0.001	0.409	0.713	0.085
AUC	-2.085	-1.257	-1.511	-4.457	-2.197	-4.029	-4.203	-0.927	-3.198
	0.037	0.209	0.131	0.001	0.028	0.001	0.001	0.354	0.001
Logarithmic	-3.595	-2.426	-4.280	-4.457	-3.340	-4.254	-4.026	-4.051	-3.215
loss	0.001	0.015	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Squared error	-3.608	-2.313	-3.975	-3.24	-2.121	-4.127	-3.518	-3.213	-2.839
	0.001	0.021	0.001	0.001	0.034	0.001	0.001	0.001	0.005
CAL score	-2.032	-1.867	-4.026	-2.806	-0.063	-4.076	-1.892	-3.543	-1.443
	0.042	0.062	0.001	0.005	0.949	0.001	0.058	0.001	0.149

Table 11: Two-sided Wilcoxon paired-samples signed ranks test comparing the performance of ISMB with other algorithms. For each performance measure the number on top is the Z statistic and the number at the bottom is the corresponding p-value. The Z statistic is negative when ISMB has a lower score on a performance measure than the competing algorithm. On all measures, a negative Z statistic indicates better performance by ISMB. Underlined results indicate p-values of 0.05 or smaller.

Performance measure	ISMB-MS	NISMB	NB	DT	LR	NN	*k*NN	LBR	AB
Misclassification error	-0.004 0.077	-0.001 0.312	-0.021 0.014	-0.013 0.021	-0.012 0.065	-0.019 <0.001	-0.005 0.334	-0.007 0.289	-0.003 0.258
AUC	-0.001 0.077	-0.002 0.242	-0.001 0.975	-0.104 <0.001	-0.017 0.022	-0.032 <0.001	-0.023 <0.001	-0.001 0.932	-0.020 0.001
Logarithmic loss	-0.009 0.001	-0.004 0.026	-0.163 0.005	-0.211 <0.001	-0.215 0.006	-0.306 <0.001	-0.140 <0.001	-0.071 <0.001	-0.103 0.002
Squared error	-0.004 0.003	-0.001 0.054	-0.044 0.002	-0.044 <0.001	-0.034 0.009	-0.062 0.002	-0.023 <0.001	-0.019 0.017	-0.016 0.007
CAL score	-0.003 0.044	-0.002 0.058	-0.033 <0.001	-0.011 0.018	-0.003 0.441	-0.047 <0.001	-0.008 0.079	-0.016 0.001	-0.002 0.237

Table 12: Two-sided paired-samples t test comparing the performance of ISMB with other algorithms. For each performance measure the number on top is the mean difference between ISMB and the indicated algorithm and the number at the bottom is the corresponding p-value. The mean difference is negative when ISMB has a lower score on a performance measure than the competing algorithm. On all measures, a negative mean difference indicates better performance by ISMB. Underlined results indicate p-values of 0.05 or smaller.

Algorithm	Average running time
NB	$<$ 1 second
DT	$<$ 1 second
LR	$<$ 1 second
NN	$<$ 1 second
kNN	$<$ 1 second
LBR	$\approx$ 1 second
AB	$<$ 1 second
ISMB	$\approx$ 2 minutes

Table 13: Approximate running times of the various algorithms. For each algorithm, the time shown is the average running time over all the UCI data sets. For the instance-specific algorithms LBR and ISMB the reported running time is for a single test instance, while for the other algorithms the reported running time is over all test instances.

Acknowledgments

This work was supported by a grant from the National Library of Medicine (NLM R01-LM008374) and a training grant from the National Library of Medicine to the University of Pittsburgh's Biomedical Informatics Training Program (T15-LM007059).

References

D. W. Aha. Feature weighting for lazy learning algorithms. In L. Huan and M. Hiroshi, editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, pages 13–32. Kluwer Academic Publisher, Norwell, MA, 1998.

C. F. Aliferis, I. Tsamardinos, and A. Statnikov. Hiton: A novel markov blanket algorithm for optimal variable selection. In *Proceedings of the American Medical Informatics Association (AMIA) Annual Symposium*, pages 21–5, 2003.

C. F. Aliferis, A. Statnikov, I. Tsamardinos, S Mani, and X. D. Koutsoukos. Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. *Journal of Machine Learning Research*, 11(Jan):171–234, 2010a.

C. F. Aliferis, A. Statnikov, I. Tsamardinos, S Mani, and X. D. Koutsoukos. Local causal and markov blanket induction for causal discovery and feature selection for classification part ii: Analysis and extensions. *Journal of Machine Learning Research*, 11(Jan):235–284, 2010b.

C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

R. Caruana and N-M. Alexandru. Data mining in metric space: An empirical analysis of supervised learning performance criteria. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–78, Seattle, WA, 2004. ACM Press.

J. Cerquides and R. Mantaras. Robust bayesian linear classifier ensembles. In *Machine Learning: ECML 2005*, volume 3720 of *Lecture Notes in Computer Science*, pages 72–83. Springer Berlin / Heidelberg, 2005.

G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

T. M. Cover and A. T. Joy. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

B. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991.

D. Dash and G. F. Cooper. Exact model averaging with naive bayesian classifiers. In C. Sammut and A. Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 91–98, Sydney, Australia, 2002. Morgan Kaufmann.

D. Dash and G. F. Cooper. Model averaging for prediction with discrete bayesian networks. *Journal of Machine Learning Research*, 5(Sep):1177–1203, 2004.

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1022–1027, Chambry, France, 1993. Morgan Kaufmann.

A. Frank and A. Asuncion. Uci machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

J. H. Friedman, R. Kohavi, and Y. Yun. Lazy decision trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 717–724, Portland, Oregon, 1996. AAAI Press.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29 (2-3):131–163, 1997.

N. Friedman, I. Nachman, and D. Pe'er. Learning bayesian network structure from massive datasets: The 'sparse-candidate' algorithm. In K. B. Laskey and H. Prade, editors, *Proceedings of the Fifteenth Annual Conference in Uncertainty in Artificial Intelligence*, pages 206–215, Stockholm, Sweden, 1999. Morgan Kaufmann.

S. Fu and M. Desmarais. Tradeoff analysis of different markov blanket local learning approaches. In *PAKDD'08: Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, pages 562–571, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-68124-8, 978-3-540-68124-3.

C. Gottrup, K. Thomsen, P. Locht, O. Wu, A. G. Sorensen, W. J. Koroshetz, and L. Ostergaard. Applying instance-based techniques to prediction of final outcome in acute stroke. *Artificial Intelligence in Medicine*, 33(3):223–236, 2005.

D. J. Hand and R. J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.

D. Heckerman. A tutorial on learning with bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks - the combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–401, 1999.

K. B. Hwang and B. T. Zhang. Bayesian model averaging of bayesian network classifiers over multiple node-orders: application to sparse datasets. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 35(6):1302–10, 2005.

R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In E. Simoudis, J. Han, and U. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, Portland, Oregon, 1996. AAAI Press.

D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292, 1996.

M. G. Madden. A new bayesian network structure for classification tasks. In *AICS '02: Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science*, pages 203–208, London, UK, 2002a. Springer-Verlag.

M. G. Madden. Evaluation of the performance of the markov blanket bayesian classifier algorithm. *CoRR*, cs.LG/0211003, 2002b.

D. Madigan and A. E. Raftery. Model selection and accounting for model uncertainty in graphical models using occam's window. *Journal of the American Statistical Association*, 89:1335–1346, 1994.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In S. A. Solla, T. K. Leen, and K.-R. Mller, editors, *Proceedings of the 1999 Conference on Advances in Neural Information Processing Systems*, Denver, CO, 1999. MIT Press.

T. P. Minka. Bayesian model averaging is not model combination. Technical report, MIT Media Lab, 2002.

A. Moore and W. K. Wong. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning*, pages 552–559. AAAI Press, 2003.

R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, Upper Saddle River, New Jersey, 1st edition, 2003.

M. J. Pazzani. Searching for dependencies in bayesian classifiers. In D. Fisher and H. J. Lenz, editors, *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 239–248, Fort Lauderdale, Florida, 1995. Springer-Verlag.

M. J. Pazzani. Constructive induction of cartesian product attributes. In L. Huan and M. Hiroshi, editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic Publisher, Norwell, MA, 1998.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, California, 1988.

A. E. Raftery, D. Madigan, and J. A. Hoeting. Model selection and accounting for model uncertainty in linear regression models. *Journal of the American Statistical Association*, 92:179–191, 1997.

K. M. Ting, Z. Zheng, and G. I. Webb. Learning lazy rules to improve the performance of classifiers. In *Proceedings of the Nineteenth SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, pages 122–131, Cambridge, UK, 1999. Springer-Verlag.

I. Tsamardinos and C. Aliferis. Towards principled feature selection: Relevancy, filters and wrappers. In Christopher M. Bishop and Brendan J. Frey, editors, *Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, FL, USA, 2003.

I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

S. Visweswaran and G. F. Cooper. Instance-specific bayesian model averaging for classification. In *Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, 2004.

S. Visweswaran and G. F. Cooper. Counting markov blanket structures. Technical Report DBMI-09-12, University of Pittsburgh, 2009.

S. Visweswaran, D. C. Angus, M. Hsieh, L. Weissfeld, D. Yealy, and G. F. Cooper. Learning patient-specific predictive models from clinical data. *Journal of Biomedical Informatics*, 43(5):669–85, 2010.

L. Wasserman. Bayesian model selection and model averaging. *Journal of Mathematical Psychology*, 44(1):92–107, 2000.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, 2005.

K. Y. Yeung, R. E. Bumgarner, and A. E. Raftery. Bayesian model averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. *Bioinformatics*, 21(10):2394–402, 2005.

J. P. Zhang, Y. S. Yim, and J. M. Yang. Intelligent selection of instances for prediction functions in lazy learning algorithms. *Artificial Intelligence Review*, 11(1-5):175–191, 1997.

Z. J. Zheng and G. I. Webb. Lazy learning of bayesian rules. *Machine Learning*, 41(1):53–84, 2000.

Journal of Machine Learning Research 11 (2010) 3371-3408 Submitted 5/10; Published 12/10

Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion

Pascal Vincent PASCAL.VINCENT@UMONTREAL.CA
Département d'informatique et de recherche opérationnelle
Université de Montréal
2920, chemin de la Tour
Montréal, Québec, H3T 1J8, Canada

Hugo Larochelle LAROCHEH@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
10 King's College Road
Toronto, Ontario, M5S 3G4, Canada

Isabelle Lajoie ISABELLE.LAJOIE.1@UMONTREAL.CA
Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Pierre-Antoine Manzagol PIERRE-ANTOINE.MANZAGOL@UMONTREAL.CA
Département d'informatique et de recherche opérationnelle
Université de Montréal
2920, chemin de la Tour
Montréal, Québec, H3T 1J8, Canada

Editor: Léon Bottou

Abstract

We explore an original strategy for building deep networks, based on stacking layers of *denoising autoencoders* which are trained locally to denoise corrupted versions of their inputs. The resulting algorithm is a straightforward variation on the stacking of ordinary autoencoders. It is however shown on a benchmark of classification problems to yield significantly lower classification error, thus bridging the performance gap with deep belief networks (DBN), and in several cases surpassing it. Higher level representations learnt in this purely unsupervised fashion also help boost the performance of subsequent SVM classifiers. Qualitative experiments show that, contrary to ordinary autoencoders, denoising autoencoders are able to learn Gabor-like edge detectors from natural image patches and larger stroke detectors from digit images. This work clearly establishes the value of using a denoising criterion as a tractable unsupervised objective to guide the learning of useful higher level representations.

Keywords: deep learning, unsupervised feature learning, deep belief networks, autoencoders, denoising

1. Introduction

It has been a long held belief in the field of neural network research that the composition of *several levels of nonlinearity* would be key to efficiently model complex relationships between variables and to achieve better generalization performance on difficult recognition tasks (McClelland et al., 1986; Hinton, 1989; Utgoff and Stracuzzi, 2002). This viewpoint is motivated in part by knowledge

of the layered architecture of regions of the human brain such as the visual cortex, and in part by a body of theoretical arguments in its favor (Håstad, 1986; Håstad and Goldmann, 1991; Bengio and LeCun, 2007; Bengio, 2009). Yet, looking back at the history of multi-layer neural networks, their problematic non-convex optimization has for a long time prevented reaping the expected benefits (Bengio et al., 2007; Bengio, 2009) of going beyond one or two hidden layers.[1] Consequently much of machine learning research has seen progress in shallow architectures allowing for convex optimization, while the difficult problem of learning in deep networks was left dormant.

The recent revival of interest in such *deep architectures* is due to the discovery of novel approaches (Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Bengio et al., 2007; Ranzato et al., 2007; Lee et al., 2008) that proved successful at learning their parameters. Several alternative techniques and refinements have been suggested since the seminal work on deep belief networks (DBN) by Hinton et al. (2006) and Hinton and Salakhutdinov (2006). All appear however to build on the same principle that we may summarize as follows:

- Training a deep network to directly optimize only the supervised objective of interest (for example the log probability of correct classification) by gradient descent, starting from random initialized parameters, does not work very well.

- What works *much* better is to initially use a *local unsupervised criterion* to (pre)train each layer in turn, with the goal of learning to produce a useful *higher-level representation* from the lower-level representation output by the previous layer. From this starting point on, gradient descent on the supervised objective leads to much better solutions in terms of generalization performance.

Deep layered networks trained in this fashion have been shown empirically to avoid getting stuck in the kind of poor solutions one typically reaches with only random initializations. See Erhan et al. (2010) for an in depth empirical study and discussion regarding possible explanations for the phenomenon.

In addition to the supervised criterion relevant to the task, what appears to be key is using an additional *unsupervised criterion* to guide the learning at each layer. In this sense, these techniques bear much in common with the semi-supervised learning approach, except that they are useful even in the scenario where all examples are labeled, exploiting the input part of the data to regularize, thus approaching better minima of generalization error (Erhan et al., 2010).

There is yet no clear understanding of what constitutes "good" representations for initializing deep architectures or what explicit unsupervised criteria may best guide their learning. We know but a few algorithms that work well for this purpose, beginning with restricted Boltzmann machines (RBMs) (Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Lee et al., 2008), and autoencoders (Bengio et al., 2007; Ranzato et al., 2007), but also semi-supervised embedding (Weston et al., 2008) and kernel PCA (Cho and Saul, 2010).

It is worth mentioning here that RBMs (Hinton, 2002; Smolensky, 1986) and basic classical autoencoders are very similar in their functional form, although their interpretation and the procedures used for training them are quite different. More specifically, the deterministic function that maps from input to *mean hidden representation*, detailed below in Section 2.2, is the same for both models. One important difference is that deterministic autoencoders consider that *real valued*

1. There is a notable exception to this in the more specialized convolutional network architecture of LeCun et al. (1989).

mean as their hidden representation whereas stochastic RBMs sample a *binary* hidden representation from that mean. However, after their initial pretraining, the way layers of RBMs are typically used in practice when stacked in a deep neural network is by propagating these real-valued means (Hinton et al., 2006; Hinton and Salakhutdinov, 2006). This is more in line with the deterministic autoencoder interpretation. Note also that reconstruction error of an autoencoder can be seen as an approximation of the log-likelihood gradient in an RBM, in a way that is similar to the approximation made by using the Contrastive Divergence updates for RBMs (Bengio and Delalleau, 2009). It is thus not surprising that initializing a deep network by stacking autoencoders yields almost as good a classification performance as when stacking RBMs (Bengio et al., 2007; Larochelle et al., 2009a). But why is it only *almost* as good? An initial motivation of the research presented here was to find a way to bridge that performance gap.

With the autoencoder paradigm in mind, we began an inquiry into the question of what can shape a good, useful representation. We were looking for unsupervised learning principles likely to lead to the learning of feature detectors that detect important structure in the input patterns.

Section 2 walks the reader along the lines of our reasoning. Starting from the simple intuitive notion of preserving information, we present a generalized formulation of the classical autoencoder, before highlighting its limitations. This leads us in Section 3 to motivate an alternative *denoising* criterion, and derive the *denoising autoencoder* model, for which we also give a possible intuitive geometric interpretation. A closer look at the considered noise types will then allow us to derive a further extension of the base model. Section 4 discusses related preexisting works and approaches. Section 5 presents experiments that qualitatively study the feature detectors learnt by a single-layer denoising autoencoder under various conditions. Section 6 describes experiments with multi-layer architectures obtained by stacking denoising autoencoders and compares their classification performance with other state-of-the-art models. Section 7 is an attempt at turning stacked (denoising) autoencoders into practical generative models, to allow for a qualitative comparison of generated samples with DBNs. Section 8 summarizes our findings and concludes our work.

1.1 Notation

We will be using the following notation throughout the article:

- Random variables are written in upper case, for example, X.
- If X is a random vector, then its j^{th} *component* will be noted X_j.
- Ordinary vectors are written in lowercase bold. For example, a realization of a random vector X may be written $\mathbf{x}$. Vectors are considered column vectors.
- Matrices are written in uppercase bold (e.g., $\mathbf{W}$). $\mathbf{I}$ denotes the identity matrix.
- The transpose of a vector $\mathbf{x}$ or a matrix $\mathbf{W}$ is written $\mathbf{x}^T$ or $\mathbf{W}^T$ (*not* $\mathbf{x}'$ or $\mathbf{W}'$ which may be used to refer to an entirely different vector or matrix).
- We use lower case p and q to denote both probability density functions or probability mass functions according to context.
- Let X and Y two random variables with marginal probability $p(X)$ and $p(Y)$. Their joint probability is written $p(X,Y)$ and the conditional $p(X|Y)$.
- We may use the following common shorthands when unambiguous: $p(\mathbf{x})$ for $p(X=\mathbf{x})$; $p(X|\mathbf{y})$ for $p(X|Y=\mathbf{y})$ (denoting a conditional distribution) and $p(\mathbf{x}|\mathbf{y})$ for $p(X=\mathbf{x}|Y=\mathbf{y})$.

- f, g, h, will be used for ordinary functions.
- Expectation (discrete case, p is probability mass): $\mathbb{E}_{p(X)}[f(X)] = \sum_{\mathbf{x}} p(X = \mathbf{x}) f(\mathbf{x})$.
- Expectation (continuous case, p is probability density): $\mathbb{E}_{p(X)}[f(X)] = \int p(\mathbf{x}) f(\mathbf{x}) \mathrm{d}\mathbf{x}$.
- Entropy or differential entropy: $\mathbb{H}(X) = \mathbb{H}(p) = \mathbb{E}_{p(X)}[-\log p(X)]$.
- Conditional entropy: $\mathbb{H}(X|Y) = \mathbb{E}_{p(X,Y)}[-\log p(X|Y)]$.
- Kullback-Leibler divergence: $\mathbb{D}_{\mathsf{KL}}(p\|q) = \mathbb{E}_{p(X)}[\log \frac{p(X)}{q(X)}]$.
- Cross-entropy: $\mathbb{H}(p\|q) = \mathbb{E}_{p(X)}[-\log q(X)] = \mathbb{H}(p) + \mathbb{D}_{\mathsf{KL}}(p\|q)$.
- Mutual information: $\mathbb{I}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y)$.
- Sigmoid: $s(x) = \frac{1}{1+e^{-x}}$ and $s(\mathbf{x}) = (s(\mathbf{x}_1), \ldots, s(\mathbf{x}_d))^T$.
- Bernoulli distribution with mean μ: $\mathcal{B}(\mu)$. By extension for vector variables: $X \sim \mathcal{B}(\mu)$ means $\forall i, X_i \sim \mathcal{B}(\mu_i)$.

1.2 General setup

We consider the typical supervised learning setup with a training set of n (input, target) pairs $D_n = \{(\mathbf{x}^{(1)}, t^{(1)}) \ldots, (\mathbf{x}^{(n)}, t^{(n)})\}$, that we suppose to be an i.i.d. sample from an **unknown distribution** $q(X,T)$ with corresponding marginals $q(X)$ and $q(T)$. We denote $q^0(X,T)$ and $q^0(X)$ the empirical distributions defined by the samples in D_n. X is a d-dimensional random vector (typically in $\mathbb{R}^d$ or in $[0,1]^d$).

In this work we are primarily concerned with finding a new, higher-level representation Y of X. Y is a d'-dimensional random vector (typically in $\mathbb{R}^{d'}$ or in $[0,1]^{d'}$). If $d' > d$ we will talk of an *over-complete* representation, whereas it will be termed an *under-complete* representation if $d' < d$. Y may be linked to X by a deterministic or stochastic mapping $q(Y|X;\theta)$ parameterized by a vector of parameters θ.

2. What Makes a Good Representation? From Mutual Information to Autoencoders

From the outset we can give an *operational definition* of a "good" representation as one that will eventually be *useful* for addressing tasks of interest, in the sense that it will help the system quickly achieve higher performance on those tasks than if it hadn't first learned to form the representation. Based on the objective measure typically used to assess algorithm performance, this might be phrased as "A good representation is one that will yield a better performing classifier". Final classification performance will indeed typically be used to objectively compare algorithms. However, if a lesson is to be learnt from the recent breakthroughs in deep network training techniques, it is that the error signal from a single narrowly defined classification task should not be the only nor primary criterion used to *guide* the learning of representations. First because it has been shown experimentally that beginning by optimizing an unsupervised criterion, oblivious of the specific classification problem, can actually greatly help in eventually achieving superior performance for that classification problem. Second it can be argued that the capacity of humans to quickly become proficient in new tasks builds on much of what they have learnt *prior* to being faced with that task.

In this section, we begin with the simple notion of retaining information and progress to formally introduce the traditional *autoencoder* paradigm from this more general vantage point.

2.1 Retaining Information about the Input

We are interested in learning a (possibly stochastic) mapping from input X to a novel representation Y. To make this more precise, let us restrict ourselves to parameterized mappings $q(Y|X) = q(Y|X;\theta)$ with parameters θ that we want to learn.

One natural criterion that we may expect any good *representation* to meet, at least to some degree, is to retain a significant amount of information about the *input*. It can be expressed in information-theoretic terms as maximizing the mutual information $\mathbb{I}(X;Y)$ between an input random variable X and its higher level representation Y. This is the *infomax principle* put forward by Linsker (1989).

Mutual information can be decomposed into an entropy and a conditional entropy term in two different ways. A first possible decomposition is $\mathbb{I}(X;Y) = \mathbb{H}(Y) - \mathbb{H}(Y|X)$ which lead Bell and Sejnowski (1995) to their infomax approach to Independent Component Analysis. Here we will start from another decomposition: $\mathbb{I}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y)$. Since observed input X comes from an unknown distribution $q(X)$ on which θ has no influence, this makes $\mathbb{H}(X)$ an unknown constant. Thus the infomax principle reduces to:

$$\begin{aligned}\arg\max_{\theta} \mathbb{I}(X;Y) &= \arg\max_{\theta} -\mathbb{H}(X|Y) \\ &= \arg\max_{\theta} \mathbb{E}_{q(X,Y)}[\log q(X|Y)].\end{aligned}$$

Now for any distribution $p(X|Y)$ we will have

$$\mathbb{E}_{q(X,Y)}[\log p(X|Y)] \leq \underbrace{\mathbb{E}_{q(X,Y)}[\log q(X|Y)]}_{-\mathbb{H}(X|Y)}, \tag{1}$$

as can easily be shown starting from the property that for any two distributions p and q we have $\mathbb{D}_{\mathsf{KL}}(q\|p) \geq 0$, and in particular $\mathbb{D}_{\mathsf{KL}}(q(X|Y=\mathbf{y})\|p(X|Y=\mathbf{y})) \geq 0$.

Let us consider a parametric distribution $p(X|Y;\theta')$, parameterized by θ', and the following optimization:

$$\max_{\theta,\theta'} \mathbb{E}_{q(X,Y;\theta)}[\log p(X|Y;\theta')].$$

From Equation 1, we see that this corresponds to maximizing a lower bound on $-\mathbb{H}(X|Y)$ and thus on the mutual information. We would end up maximizing the *exact* mutual information provided $\exists\theta'$ s.t. $q(X|Y) = p(X|Y;\theta')$.

If, as is done in infomax ICA, we further restrict ourselves to a deterministic mapping from X to Y, that is, representation Y is to be computed by a parameterized function $Y = f_\theta(X)$ or equivalently $q(Y|X;\theta) = \delta(Y - f_\theta(X))$ (where δ denotes Dirac-delta), then this optimization can be written:

$$\max_{\theta,\theta'} \mathbb{E}_{q(X)}[\log p(X|Y = f_\theta(X);\theta')].$$

This again corresponds to maximizing a lower bound on the mutual information.

Since $q(X)$ is unknown, but we have samples from it, the empirical average over the training samples can be used instead as an unbiased estimate (i.e., replacing $\mathbb{E}_{q(X)}$ by $\mathbb{E}_{q^0(X)}$):

$$\max_{\theta,\theta'} \mathbb{E}_{q^0(X)}[\log p(X|Y = f_\theta(X);\theta')]. \tag{2}$$

We will see in the next section that this equation corresponds to the *reconstruction error* criterion used to train *autoencoders*.

2.2 Traditional Autoencoders (AE)

Here we briefly specify the traditional *autoencoder* (AE)[2] framework and its terminology, based on f_θ and $p(X|Y;\theta')$ introduced above.

Encoder: The deterministic mapping f_θ that transforms an input vector $\mathbf{x}$ into hidden representation $\mathbf{y}$ is called the **encoder**. Its typical form is an affine mapping followed by a nonlinearity:

$$f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x}+\mathbf{b}).$$

Its parameter set is $\theta = \{\mathbf{W},\mathbf{b}\}$, where $\mathbf{W}$ is a $d' \times d$ weight matrix and $\mathbf{b}$ is an offset vector of dimensionality d'.

Decoder: The resulting hidden representation $\mathbf{y}$ is then mapped back to a reconstructed d-dimensional vector $\mathbf{z}$ in input space, $\mathbf{z} = g_{\theta'}(\mathbf{y})$. This mapping $g_{\theta'}$ is called the **decoder**. Its typical form is again an affine mapping optionally followed by a squashing non-linearity, that is, either $g_{\theta'}(\mathbf{y}) = \mathbf{W}'\mathbf{y}+\mathbf{b}'$ or

$$g_{\theta'}(\mathbf{y}) = s(\mathbf{W}'\mathbf{y}+\mathbf{b}'), \tag{3}$$

with appropriately sized parameters $\theta' = \{\mathbf{W}',\mathbf{b}'\}$.

In general $\mathbf{z}$ is not to be interpreted as an exact reconstruction of $\mathbf{x}$, but rather in probabilistic terms as the parameters (typically the mean) of a distribution $p(X|Z=\mathbf{z})$ that may generate $\mathbf{x}$ with high probability. We have thus completed the specification of $p(X|Y;\theta')$ from the previous section as $p(X|Y=\mathbf{y}) = p(X|Z=g_{\theta'}(\mathbf{y}))$. This yields an associated reconstruction error to be optimized:

$$L(\mathbf{x},\mathbf{z}) \propto -\log p(\mathbf{x}|\mathbf{z}). \tag{4}$$

Common choices for $p(\mathbf{x}|\mathbf{z})$ and associated loss function $L(\mathbf{x},\mathbf{z})$ include:

- For real-valued $\mathbf{x}$, that is, $\mathbf{x} \in \mathbb{R}^d$: $X|\mathbf{z} \sim \mathcal{N}(\mathbf{z},\sigma^2\mathbf{I})$, that is, $X_j|\mathbf{z} \sim \mathcal{N}(\mathbf{z}_j,\sigma^2)$.
  This yields $L(\mathbf{x},\mathbf{z}) = L_2(\mathbf{x},\mathbf{z}) = C(\sigma^2)\|\mathbf{x}-\mathbf{z}\|^2$ where $C(\sigma^2)$ denotes a constant that depends only on σ^2 and that can be ignored for the optimization. This is the *squared error* objective found in most traditional autoencoders. In this setting, due to the Gaussian interpretation, it is more natural *not* to use a squashing nonlinearity in the decoder.

- For binary $\mathbf{x}$, that is, $\mathbf{x} \in \{0,1\}^d$: $X|\mathbf{z} \sim \mathcal{B}(\mathbf{z})$, that is, $X_j|\mathbf{z} \sim \mathcal{B}(\mathbf{z}_j)$.
  In this case, the decoder needs to produce a $\mathbf{z} \in [0,1]^d$. So a squashing nonlinearity such as a sigmoid s will typically be used in the decoder. This yields $L(\mathbf{x},\mathbf{z}) = L_{\mathbb{H}}(\mathbf{x},\mathbf{z}) = -\sum_j[\mathbf{x}_j\log\mathbf{z}_j + (1-\mathbf{x}_j)\log(1-\mathbf{z}_j)] = \mathbb{H}(\mathcal{B}(\mathbf{x})\|\mathcal{B}(\mathbf{z}))$ which is termed the *cross-entropy loss* because it is seen as the cross-entropy between two independent multivariate Bernoullis, the first with mean $\mathbf{x}$ and the other with mean $\mathbf{z}$. This loss can also be used when $\mathbf{x}$ is not strictly binary but rather $\mathbf{x} \in [0,1]^d$.

2. Note: *AutoEncoders* (AE) are also often called *AutoAssociators* (AA) in the literature. The shorter autoencoder term was preferred in this work, as we believe *encoding* better conveys the idea of producing a novel useful representation. Similarly, what we call Stacked Auto Encoders (SAE) has also been called Stacked AutoAssociators (SAA).

Note that in the general autoencoder framework, we may use other forms of parameterized functions for the encoder or decoder, and other suitable choices of the loss function (corresponding to a different $p(X|z)$). In particular, we investigated the usefulness of a more complex encoding function in Larochelle, Erhan, and Vincent (2009b). For the experiments in the present work however, we will restrict ourselves to the two usual forms detailed above, that is, an **affine+sigmoid encoder** and either **affine decoder with squared error loss** or **affine+sigmoid decoder with cross-entropy loss**. A further constraint that can optionally be imposed, and that further parallels the workings of RBMs, is having *tied weights* between W and W', in effect defining W' as $W' = W^T$.

Autoencoder training consists in minimizing the reconstruction error, that is, carrying the following optimization:

$$\arg\min_{\theta,\theta'} \mathbb{E}_{q^0(X)}[L(X, Z(X))],$$

where we wrote $Z(X)$ to emphasize the fact that Z is a deterministic function of X, since Z is obtained by composition of deterministic encoding and decoding.

Making this explicit and using our definition of loss L from Equation 4 this can be rewritten as:

$$\arg\max_{\theta,\theta'} \mathbb{E}_{q^0(X)}[\log p(X|Z = g_{\theta'}(f_\theta(X)))],$$

or equivalently

$$\arg\max_{\theta,\theta'} \mathbb{E}_{q^0(X)}[\log p(X|Y = f_\theta(X); \theta')].$$

We see that this last line corresponds to Equation 2, that is, the maximization of a lower bound on the mutual information between X and Y.

It can thus be said that **training an autoencoder to minimize reconstruction error amounts to maximizing a lower bound on the mutual information between input X and learnt representation Y**. Intuitively, if a representation allows a good reconstruction of its input, it means that it has retained much of the information that was present in that input.

2.3 Merely Retaining Information is Not Enough

The criterion that representation Y should retain information about input X is not by itself sufficient to yield a useful representation. Indeed mutual information can be trivially maximized by setting $Y = X$. Similarly, an ordinary autoencoder where Y is of the same dimensionality as X (or larger) can achieve perfect reconstruction simply by learning an identity mapping.[3] Without any other constraints, this criterion alone is unlikely to lead to the discovery of a more useful representation than the input.

Thus further constraints need to be applied to attempt to separate useful information (to be retained) from noise (to be discarded). This will naturally translate to non-zero reconstruction error. The traditional approach to autoencoders uses a *bottleneck* to produce an *under-complete* representation where $d' < d$. The resulting lower-dimensional Y can thus be seen as a *lossy compressed representation* of X. When using affine encoder and decoder without any nonlinearity and a squared error loss, the autoencoder essentially performs principal component analysis (PCA) as showed by

3. More precisely, it suffices that $g \circ f$ be the identity to obtain zero reconstruction error. For $d = d'$ if we had a linear encoder and decoder this would be achieved for any invertible matrix $\mathbf{W}$ by setting $\mathbf{W}' = \mathbf{W}^{-1}$. Now there is a sigmoid nonlinearity in the encoder, but it is possible to stay in the linear part of the sigmoid with small enough $\mathbf{W}$.

Baldi and Hornik (1989).[4] When a nonlinearity such as a sigmoid is used in the encoder, things become a little more complicated: obtaining the PCA subspace is a likely possibility (Bourlard and Kamp, 1988) since it is possible to stay in the linear regime of the sigmoid, but arguably not the only one (Japkowicz et al., 2000). Also when using a cross-entropy loss rather than a squared error the optimization objective is no longer the same as that of PCA and will likely learn different features. The use of "tied weights" can also change the solution: forcing encoder and decoder matrices to be symmetric and thus have the same scale can make it harder for the encoder to stay in the linear regime of its nonlinearity without paying a high price in reconstruction error.

Alternatively it is also conceivable to impose on Y different constraints than that of a lower dimensionality. In particular the possibility of using *over-complete* (i.e., higher dimensional than the input) but *sparse* representations has received much attention lately. Interest in sparse representations is inspired in part by evidence that neural activity in the brain seems to be sparse and has burgeoned following the seminal work of Olshausen and Field (1996) on *sparse coding*. Other motivations for sparse representations include the ability to handle effectively variable-size representations (counting only the non-zeros), and the fact that dense compressed representations tend to entangle information (i.e., changing a single aspect of the input yields significant changes in all components of the representation) whereas sparse ones can be expected to be easier to interpret and to use for a subsequent classifier. Various modifications of the traditional autoencoder framework have been proposed in order to learn sparse representations (Ranzato et al., 2007, 2008). These were shown to extract very useful representations, from which it is possible to build top performing deep neural network classifiers. A sparse over-complete representations can be viewed as an alternative "compressed" representation: it has *implicit* straightforward compressibility due to the large number of zeros rather than an explicit lower dimensionality.

3. Using a Denoising Criterion

We have seen that the reconstruction criterion alone is unable to guarantee the extraction of useful features as it can lead to the obvious solution "simply copy the input" or similarly uninteresting ones that trivially maximizes mutual information. One strategy to avoid this phenomenon is to constrain the representation: the traditional bottleneck and the more recent interest on sparse representations both follow this strategy.

Here we propose and explore a very different strategy. Rather than constrain the representation, we change the reconstruction criterion for a both more challenging and more interesting objective: cleaning partially corrupted input, or in short *denoising*. In doing so we modify the implicit definition of a good representation into the following: *"a good representation is one that can be obtained robustly from a corrupted input and that will be useful for recovering the corresponding clean input"*. Two underlying ideas are implicit in this approach:

- First it is expected that a higher level representation should be rather stable and robust under corruptions of the input.
- Second, it is expected that performing the denoising task well requires extracting features that capture useful structure in the input distribution.

4. More specifically it will find the same *subspace* as PCA, but the specific projection directions found will in general not correspond to the actual principal directions and need not be orthonormal.

We emphasize here that our goal is *not* the task of denoising per se. Rather **denoising is advocated and investigated as a *training criterion* for learning to extract useful features** that will constitute better higher level representation. The usefulness of a learnt representation can then be assessed objectively by measuring the accuracy of a classifier that uses it as input.

3.1 The Denoising Autoencoder Algorithm

This approach leads to a very simple variant of the basic autoencoder described above. A *denoising autoencoder (DAE)* is trained to reconstruct a clean "repaired" input from a *corrupted* version of it (the specific types of corruptions we consider will be discussed below). This is done by first corrupting the initial input $\mathbf{x}$ into $\tilde{\mathbf{x}}$ by means of a stochastic mapping $\tilde{\mathbf{x}} \sim q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x})$.

Corrupted input $\tilde{\mathbf{x}}$ is then mapped, as with the basic autoencoder, to a hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}}) = s(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b})$ from which we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y})$. See Figure 1 for a schematic representation of the procedure. Parameters θ and θ' are trained to minimize the average reconstruction error over a training set, that is, to have $\mathbf{z}$ as close as possible to the *uncorrupted* input $\mathbf{x}$. The key difference is that $\mathbf{z}$ is now a deterministic function of $\tilde{\mathbf{x}}$ rather than $\mathbf{x}$. As previously, the considered reconstruction error is either the cross-entropy loss $L_{\mathbb{H}}(\mathbf{x},\mathbf{z}) = \mathbb{H}(\mathcal{B}(\mathbf{x})\|\mathcal{B}(\mathbf{z}))$, with an affine+sigmoid decoder, or the squared error loss $L_2(\mathbf{x},\mathbf{z}) = \|\mathbf{x}-\mathbf{z}\|^2$, with an affine decoder. Parameters are initialized at random and then optimized by stochastic gradient descent. Note that each time a training example $\mathbf{x}$ is presented, a different corrupted version $\tilde{\mathbf{x}}$ of it is generated according to $q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x})$.

Note that denoising autoencoders are still minimizing the same reconstruction loss between a clean X and its reconstruction from Y. So this still amounts to maximizing a lower bound on the mutual information between clean input X and representation Y. The difference is that Y is now obtained by applying deterministic mapping f_θ to a *corrupted* input. It thus forces the learning of a far more clever mapping than the identity: one that extracts features useful for *denoising*.

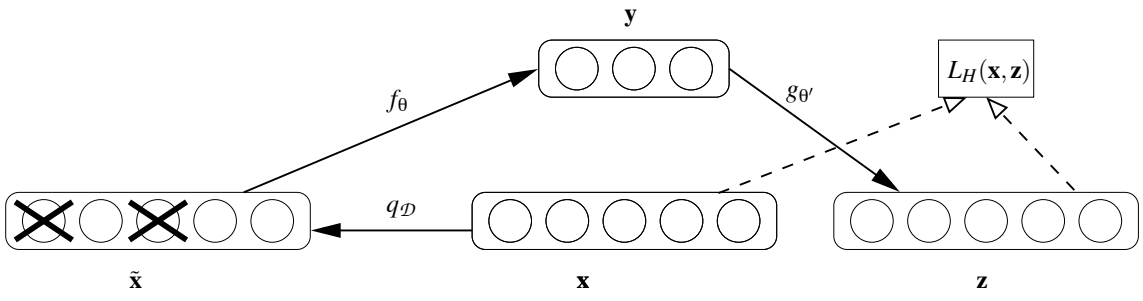


Figure 1: The denoising autoencoder architecture. An example $\mathbf{x}$ is stochastically corrupted (via $q_{\mathcal{D}}$) to $\tilde{\mathbf{x}}$. The autoencoder then maps it to $\mathbf{y}$ (via encoder f_θ) and attempts to reconstruct $\mathbf{x}$ via decoder $g_{\theta'}$, producing reconstruction $\mathbf{z}$. Reconstruction error is measured by loss $L_H(\mathbf{x},\mathbf{z})$.

3.2 Geometric Interpretation

The process of denoising, that is, mapping a corrupted example back to an uncorrupted one, can be given an intuitive geometric interpretation under the so-called *manifold assumption* (Chapelle et al., 2006), which states that natural high dimensional data concentrates close to a non-linear low-dimensional manifold. This is illustrated in Figure 2. During denoising training, we learn a stochastic operator $p(X|\widetilde{X})$ that maps a corrupted $\widetilde{X}$ back to its uncorrupted X, for example, in the case of binary data,

$$X|\widetilde{X} \sim \mathcal{B}(g_{\theta'}(f_\theta(\widetilde{X}))).$$

Corrupted examples are much more likely to be outside and farther from the manifold than the uncorrupted ones. Thus stochastic operator $p(X|\widetilde{X})$ learns a map that tends to go from lower probability points $\widetilde{X}$ to nearby high probability points X, on or near the manifold. Note that when $\widetilde{X}$ is farther from the manifold, $p(X|\widetilde{X})$ should learn to make bigger steps, to reach the manifold. Successful denoising implies that the operator maps even far away points to a small region close to the manifold.

The denoising autoencoder can thus be seen as a way to define and learn a manifold. In particular, if we constrain the dimension of Y to be smaller than the dimension of X, then the intermediate representation $Y = f(X)$ may be interpreted as a coordinate system for points on the manifold. More generally, one can think of $Y = f(X)$ as a representation of X which is well suited to capture the main variations in the data, that is, those along the manifold.

3.3 Types of Corruption Considered

The above principle and technique can potentially be used with any type of corruption process. Also the corruption process is an obvious place where prior knowledge, if available, could be easily incorporated. But in the present study we set to investigate a technique that is generally applicable. In

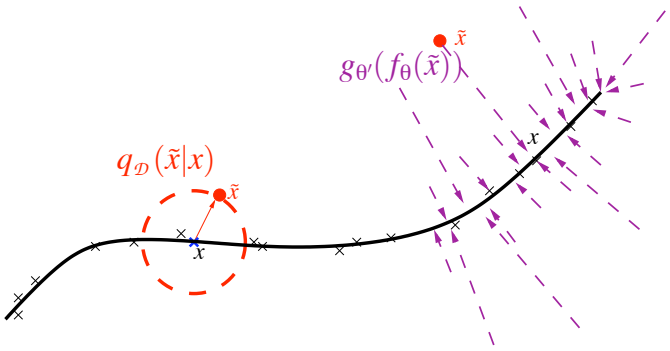


Figure 2: Manifold learning perspective. Suppose training data ($\times$) concentrate near a low-dimensional manifold. Corrupted examples ($\bullet$) obtained by applying corruption process $q_{\mathcal{D}}(\widetilde{X}|X)$ will generally lie farther from the manifold. The model learns with $p(X|\widetilde{X})$ to "project them back" (via autoencoder $g'_\theta(f_\theta(\cdot))$) onto the manifold. Intermediate representation $Y = f_\theta(X)$ may be interpreted as a coordinate system for points X on the manifold.

particular we want it to be usable for learning ever higher level representations by *stacking* denoising autoencoders. Now while prior knowledge on relevant corruption processes may be available in a particular input space (such as images), such prior knowledge will not be available for the space of intermediate-level representations.

We will thus restrict our discussion and experiments to the following simple corruption processes:

- Additive isotropic *Gaussian noise* (GS): $\tilde{\mathbf{x}}|\mathbf{x} \sim \mathcal{N}(\mathbf{x}, \sigma^2 I)$;
- *Masking noise* (MN): a fraction ν of the elements of $\mathbf{x}$ (chosen at random for each example) is forced to 0;
- *Salt-and-pepper noise* (SP): a fraction ν of the elements of $\mathbf{x}$ (chosen at random for each example) is set to their minimum or maximum possible value (typically 0 or 1) according to a fair coin flip.

Additive Gaussian noise is a very common noise model, and is a natural choice for real valued inputs. The *salt-and-pepper noise* will also be considered, as it is a natural choice for input domains which are interpretable as binary or near binary such as black and white images or the representations produced at the hidden layer after a sigmoid squashing function.

Much of our work however, both for its inspiration and in experiments, focuses on *masking noise* which can be viewed as turning off components considered missing or replacing their value by a default value—that is, a common technique for handling missing values. All information about these masked components is thus removed from that particular input pattern, and we can view the denoising autoencoder as trained to *fill-in* these artificially introduced "blanks". Also, numerically, forcing components to zero means that they are totally ignored in the computations of downstream neurons.

We draw the reader's attention to the fact that both salt-and-pepper and masking noise drastically corrupt but a fraction of the elements while leaving the others untouched. Denoising, that is, recovering the values of the corrupted elements, will only be possible thanks to dependencies between dimensions in high dimensional distributions. Denoising training is thus expected to capture these dependencies. The approach probably makes less sense for very low dimensional problems, at least with these types of corruption.

3.4 Extension: Putting an Emphasis on Corrupted Dimensions

Noise types such as *masking noise* and *salt-and-pepper* that erase only a changing subset of the input's components while leaving the others untouched suggest a straightforward extension of the denoising autoencoder criterion. Rather than giving equal weight to the reconstruction of all components of the input, we can put an *emphasis* on the corrupted dimensions. To achieve this we give a different weight α for the reconstruction error on components that were corrupted, and β for those that were left untouched. α and β are considered hyperparameters.

For the squared loss this yields

$$L_{2,\alpha}(\mathbf{x}, \mathbf{z}) = \alpha \left(\sum_{j \in \mathcal{J}(\tilde{\mathbf{x}})} (\mathbf{x}_j - \mathbf{z}_j)^2 \right) + \beta \left(\sum_{j \notin \mathcal{J}(\tilde{\mathbf{x}})} (\mathbf{x}_j - \mathbf{z}_j)^2 \right),$$

where $\mathcal{J}(\tilde{\mathbf{x}})$ denotes the indexes of the components of $\mathbf{x}$ that were corrupted.

And for the cross-entropy loss this yields

$$L_{\mathsf{IH},\alpha}(\mathbf{x},\mathbf{z}) = \alpha\left(-\sum_{j\in\mathcal{J}(\tilde{\mathbf{x}})}[\mathbf{x}_j\log\mathbf{z}_j+(1-\mathbf{x}_j)\log(1-\mathbf{z}_j)]\right) + \beta\left(-\sum_{j\notin\mathcal{J}(\tilde{\mathbf{x}})}[\mathbf{x}_j\log\mathbf{z}_j+(1-\mathbf{x}_j)\log(1-\mathbf{z}_j)]\right).$$

We call this extension *emphasized denoising autoencoder*. A special case that we call *full emphasis* is obtained for $\alpha = 1,\ \beta = 0$ where we only take into account the error on the prediction of corrupted elements.

3.5 Stacking Denoising Autoencoders to Build Deep Architectures

Stacking denoising autoencoders to initialize a deep network works in much the same way as stacking RBMs in deep belief networks (Hinton et al., 2006; Hinton and Salakhutdinov, 2006) or ordinary autoencoders (Bengio et al., 2007; Ranzato et al., 2007; Larochelle et al., 2009a). Let us specify that input corruption is only used for the initial denoising-training of each individual layer, so that it may learn useful feature extractors. Once the mapping f_θ has thus been learnt, it will henceforth be used on *uncorrupted* inputs. In particular no corruption is applied to produce the representation that will serve as clean input for training the next layer. The complete procedure for learning and stacking several layers of denoising autoencoders is shown in Figure 3.

Once a stack of encoders has thus been built, its highest level output representation can be used as input to a stand-alone supervised learning algorithm, for example a Support Vector Machine classifier or a (multi-class) logistic regression. Alternatively, as illustrated in Figure 4, a logistic regression layer can be added on top of the encoders, yielding a *deep neural network* amenable to supervised learning. The parameters of all layers can then be simultaneously *fine-tuned* using a gradient-based procedure such as stochastic gradient descent.

4. Related Approaches in the Literature

In this section, we briefly review and discuss related prior work along three different axes.

4.1 Previous Work on Training Neural Networks for Denoising

The idea of training a multi-layer perceptron using error backpropagation on a denoising task is not new. The approach was first introduced by LeCun (1987) and Gallinari et al. (1987) as an alternative method to learn an *(auto-)associative memory* similar to how Hopfield Networks (Hopfield, 1982) were understood. The networks were trained and tested on binary input patterns, corrupted by flipping a fraction of input bits chosen at random. Both the model and training procedure in this precursory work are very similar to the denoising autoencoder we describe.[5] Our motivation and goal are however quite different. The objective of LeCun (1987) was to study the *capacity* of such a network for memorization tasks, that is, counting how many training patterns it was able to

5. There are a few minor differences; for example, the use of a squared error after sigmoid for binary data, while we tend to use a cross-entropy loss. Also their denoising procedure considers doing several recurrent passes through the autoencoder network, as in a recurrent net.

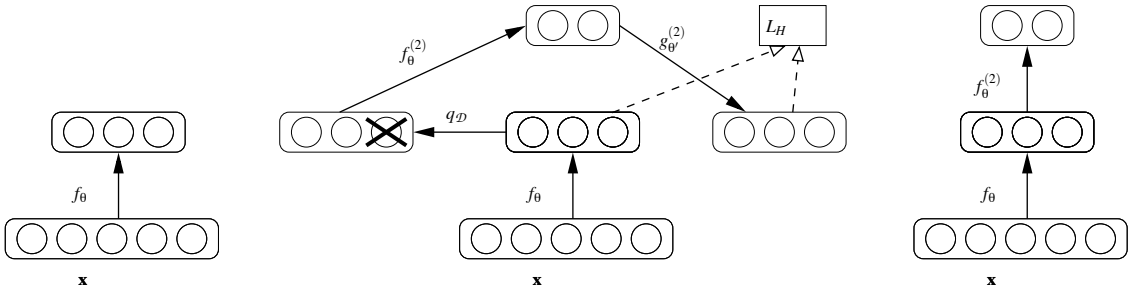


Figure 3: Stacking denoising autoencoders. After training a first level denoising autoencoder (see Figure 1) its learnt encoding function f_θ is used on clean input (left). The resulting representation is used to train a second level denoising autoencoder (middle) to learn a second level encoding function $f_\theta^{(2)}$. From there, the procedure can be repeated (right).

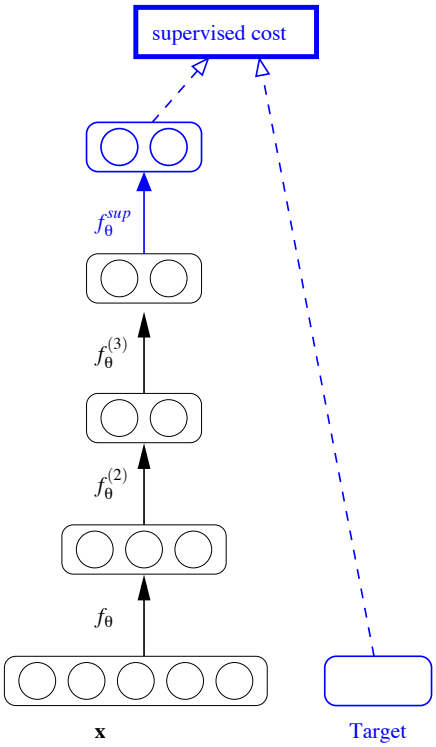


Figure 4: Fine-tuning of a deep network for classification. After training a stack of encoders as explained in the previous figure, an output layer is added on top of the stack. The parameters of the whole system are fine-tuned to minimize the error in predicting the supervised target (e.g., class), by performing gradient descent on a supervised cost.

correctly recall under these conditions. The work also clearly established the usefulness of a non-linear hidden layer for this. By contrast, our work is motivated by the search and understanding of unsupervised pretraining criteria to initialize deep networks. Our primary interest is thus in investigating the ability of the denoising criterion to learn good feature extractors, with which to initialize a deep network by stacking and composing these feature extractors. We focus on analyzing the learnt higher-level representations and their effect on the classification performance of resulting deep networks.

Another insightful work that is very much related to the approach advocated here is the research of Seung (1998), in which a recurrent neural network is trained to complete corrupted input patterns using backpropagation through time. Both the work of Seung (1998) and that of LeCun (1987) and Gallinari et al. (1987) appear to be inspired by Hopfield-type associative memories (Hopfield, 1982) in which learnt patterns are conceived as attractive fixed points of a recurrent network dynamic. Seung (1998) contributes a very interesting analysis in terms of continuous attractors, points out the limitations of regular autoencoding, and advocates the pattern completion task as an alternative to density estimation for unsupervised learning. Again, it differs form our study mainly by its *focus* a) on recurrent networks[6] and b) on the image denoising task per se. The latter justifies their use of prior knowledge of the 2D topology of images, both in the architectural choice of local 2D receptive field connectivity, and in the corruption process that consists in zeroing-out a square image patch at a random position. This occlusion by a 2D patch is a special form of a *structured* masking noise, where the a-priori known 2D topological structure of images is taken into account. In our research we deliberately chose not to use topological prior knowledge in our model nor in our corruption process, so that the *same generic procedure* may be applied to learn higher levels of representation from lower ones, or to other domains for which we have no such topological prior knowledge.

More recently Jain and Seung (2008) presented a very interesting and successful approach for image denoising, that consists in layer-wise building of a deep convolutional neural network. Their algorithm yields comparable or better performance than state-of-the-art Markov Random Field and wavelet methods developed for image denoising. The approach clearly has roots in their earlier work (Seung, 1998) and appears also inspired by more recent research on deep network pretraining, including our own group's (Bengio et al., 2007). But apparently, neither of us was initially aware of the other group's relevant work on denoising (Vincent et al., 2008; Jain and Seung, 2008). Again the focus of Seung (1998) on image denoising per se differs from our own focus on studying deep network pretraining for classification tasks and results in marked differences in the actual algorithms. Specifically, in Jain and Seung (2008) each layer in the stack is trained to reconstruct the original clean image a little better, which makes sense for image denoising. This can be contrasted with our approach, in which upper layers are trained to denoise-and-reconstruct whatever representation they receive from the layer immediately below, rather than to restore the original input image in one operation. This logically follows from our search for a generic feature extraction algorithm for pretraining, where upper level representations will eventually be used for a totally different task such as classification.

6. Note however that a recurrent network can be seen as deep network, with the additional property that all layers share the same weights.

4.2 Training Classifiers with Noisy Inputs

The idea of training a neural network with noisy input (Scalettar and Zee, 1988; von Lehman et al., 1988)—or training with *jitter* as it is sometimes called—has been proposed to enhance generalization performance for supervised learning tasks (Sietsma and Dow, 1991; Holmstrm and Koistinen, 1992; An, 1996). This thread of research is less directly related to autoencoders and denoising than the studies discussed in the previous section. It is nevertheless relevant. After all, denoising amounts to using noisy patterns as input with the clean pattern as a supervised target, albeit a rather high dimensional one. It has been argued that training with noise is equivalent to applying generalized Tikhonov regularization (Bishop, 1995). On the surface, this may seem to suggest that training with noisy inputs has a similar effect to training with an L2 weight decay penalty (i.e., penalizing the sum of squared weights), but this view is incorrect. Tikhonov regularization applied to *linear regression* is indeed equivalent to a L2 weight decay penalty (i.e., ridge regression). But for a non-linear mapping such as a neural network, Tikhonov regularization is no longer so simple (Bishop, 1995). More importantly, in the non-linear case, the equivalence of noisy training with Tikhonov regularization is derived from a Taylor series expansion, and is thus only valid in the limit of very small additive noise. See Grandvalet et al. (1997) for a theoretical study and discussion regarding the limitations of validity for this equivalence. Last but not least, our experimental results in Section 5.1 clearly show qualitatively very different results when using denoising autoencoders (i.e., noisy inputs) than when using regular autoencoders with a L2 weight decay.

Here, we must also mention a well-known technique to improve the generalization performance of a classifier (neural network or other), which consists in augmenting the original training set with additional distorted inputs, either explicitly (Baird, 1990; Poggio and Vetter, 1992) or virtually through a modified algorithm (Simard et al., 1992; Schölkopf et al., 1996). For character images for instance such distortions may include small translations, rotations, scalings and shearings of the image, or even applying a scanner noise model. This technique can thus be seen as training with noisy corrupted inputs, but with a highly structured corruption process based on *much* prior knowledge.[7] As already explained and motivated above, our intent in this work is to develop and investigate a generally applicable technique, that should also be applicable to intermediate higher level representations. Thus we intentionally restrict our study to very simple generic corruption processes that do not incorporate explicit prior knowledge.

We also stress the difference between the approaches we just discussed, that consist in training a neural network by optimizing a *global supervised criterion using noisy input*, and the approach we investigate in the present work, that is, using a *local unsupervised denoising criterion to pretrain each layer of the network* with the goal to learn useful intermediate representations. We shall see in experimental Section 6.4 that the latter applied to a deep network yields better classification performance than the former.

4.3 Pseudo-Likelihood and Dependency Networks

The view of denoising training as "filling in the blanks" that motivated the masking noise and the extension that puts an emphasis on corrupted dimensions presented in Section 3.4, can also be related to the pseudo-likelihood (Besag, 1975) and Dependency Network (Heckerman et al., 2000)

7. Clearly, simple Tikhonov regularization cannot achieve the same as training with such prior knowledge based corruption process. This further illustrates the limitation of the equivalence between training with noise and Tikhonov regularization.

paradigms. Maximizing pseudo-likelihood instead of likelihood implies replacing the likelihood term $p(X)$ by the product of conditionals $\prod_{i=1}^{d} p(X_i|X_{\neg i})$. Here X_i denotes the i^{th} component of input vector variable X and $X_{\neg i}$ denotes all components but the i^{th}. Similarly, in the Dependency Network approach of Heckerman et al. (2000) one learns d conditional distributions, each trained to predict component i given (a subset of) all other components. This is in effect what an *emphasized denoising autoencoder* with a masking noise that masks but one input component ($\nu = \frac{1}{d}$), and a *full emphasis* ($\alpha = 1,\ \beta = 0$), is trained to do. More specifically, for binary variables it will learn to predict $p(X_i = 1|X_{\neg i})$; and when using squared error for real-valued variables it will learn to predict $\mathbb{E}[X_i|X_{\neg i}]$ assuming $X_i|X_{\neg i} \sim \mathcal{N}(\mathbb{E}[X_i|X_{\neg i}], \sigma^2)$. Note that with denoising autoencoders, all d conditional distributions are constrained to *share common parameters*, which define the mapping to and from hidden representation Y. Also when the emphasis is not fully put on the corrupted components ($\beta > 0$) some of the network's capacity will be devoted to encoding/decoding uncorrupted components.

A more important difference can be appreciated by considering the following scenario: What happens if components of the input come in identical pairs? In that case, conditional distribution $p(X_i|X_{\neg i})$ can simply learn to replicate the other component of the pair, thus not capturing any other potentially useful dependency. Now for dependency networks this exact scenario is forbidden by the formal requirement that no input configuration may have zero probability. But a similar problem may occur in practice if the components in a pair are not identical but very highly correlated. By contrast, denoising autoencoders can and will typically be trained with a larger fraction ν of corrupted components, so that reliable prediction of a component cannot rely exclusively on a single other component.

5. Experiments on Single Denoising Autoencoders: Qualitative Evaluation of Learned Feature Detectors

A first series of experiments was carried out using single denoising autoencoders, that is, without any stacking nor supervised fine tuning. The goal was to examine qualitatively the kind of feature detectors learnt by a denoising criterion, for various noise types, and compare these to what ordinary autoencoders yield.

Feature detectors that correspond to the first hidden layer of a network trained on image data are straightforward to visualize. Each hidden neuron $\mathbf{y}_j$ has an associated vector of weights $\mathbf{W}_j$ that it uses to compute a dot product with an input example (before applying its non-linearity). These $\mathbf{W}_j$ vectors, the *filters*, have the same dimensionality as the input, and can thus be displayed as little images showing what aspects of the input each hidden neuron is sensitive to.

5.1 Feature Detectors Learnt from Natural Image Patches

We trained both regular autoencoders and denoising autoencoders on 12×12 patches from whitened natural scene images, made available by Olshausen (Olshausen and Field, 1996).[8] A few of these patches are shown in Figure 5 (left). For these natural image patches, we used a linear decoder and a squared reconstruction cost. Network parameters were trained from a random start,[9] using

8. More specifically randomly positioned 12×12 patches were extracted from the 20×20 patches made available by Olshausen at the following URL: `https://redwood.berkeley.edu/bruno/sparsenet/`.
9. We applied the usual random initialization heuristic in which weights are sampled independently form a uniform in range $[-\frac{1}{\sqrt{\text{fanin}}}, \frac{1}{\sqrt{\text{fanin}}}]$ where fanin in this case is the input dimension.

stochastic gradient descent to perform 500000 weight updates with a fixed learning rate of 0.05. All filters shown were from experiments with tied weights, but untied weights yielded similar results.

Figure 5 displays filters learnt by a regular *under-complete* autoencoder that used a bottleneck of 50 hidden units, as well as those learnt by an *over-complete* autoencoder using 200 hidden units. Filters obtained in the under-complete case look like very local blob detectors. No clear structure is apparent in the filters learnt in the over-complete case.

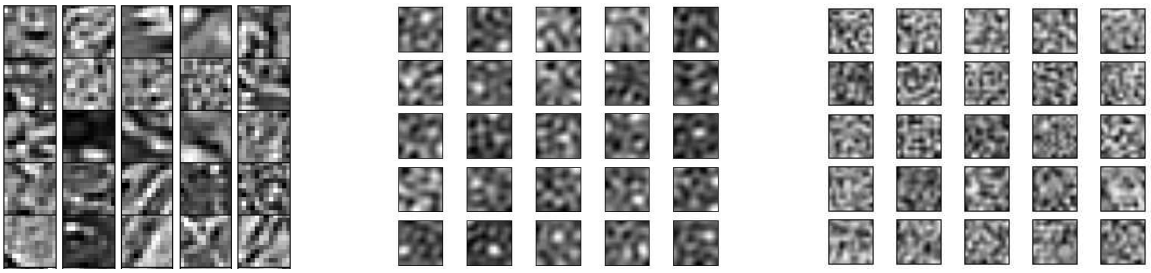


Figure 5: Regular autoencoder trained on natural image patches. *Left:* some of the 12×12 image patches used for training. *Middle:* filters learnt by a regular *under-complete* autoencoder (50 hidden units) using tied weights and L2 reconstruction error. *Right:* filters learnt by a regular *over-complete* autoencoder (200 hidden units). The under-complete autoencoder appears to learn rather uninteresting local blob detectors. Filters obtained in the over-complete case have no recognizable structure, looking entirely random.

We then trained 200 hidden units over-complete noiseless autoencoders regularized with L2 weight decay, as well as 200 hidden units denoising autoencoders with isotropic Gaussian noise (but no weight decay). Resulting filters are shown in Figure 6. Note that a denoising autoencoder with a noise level of 0 is identical to a regular autoencoder. So, naturally, filters learnt by a denoising autoencoder at small noise levels (not shown) look like those obtained with a regular autoencoder previously shown in Figure 5. With a sufficiently large noise level however ($\sigma = 0.5$), the denoising autoencoder learns Gabor-like local oriented edge detectors (see Figure 6). This is similar to what is learnt by sparse coding (Olshausen and Field, 1996, 1997), or ICA (Bell and Sejnowski, 1997) and resembles simple cell receptive fields from the primary visual cortex first studied by Hubel and Wiesel (1959). The L2 regularized autoencoder on the other hand learnt nothing interesting beyond restoring some of the local blob detectors found in the under-complete case. Note that we did try a wide range of values for the regularization hyperparameter,[10] but were never able to get Gabor-like filters. From this experiment, we see clearly that **training with sufficiently large noise yields a qualitatively very different outcome than training with a weight decay regularization**, which confirms experimentally that the two are *not* equivalent for a non-linear autoencoder, as discussed earlier in Section 4.2.

Figure 7 shows some of the results obtained with the other two noise types considered, that is, salt-and-pepper noise, and masking-noise. We experimented with 3 corruption levels ν: $10\%, 25\%, 55\%$. The filters shown were obtained using 100 hidden units, but similar filters were found with 50 or 200 hidden units. Salt-and-pepper noise yielded Gabor-like edge detectors, whereas masking noise

10. Attempted weight decays values were the following: $\lambda \in \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 1.0\}$.

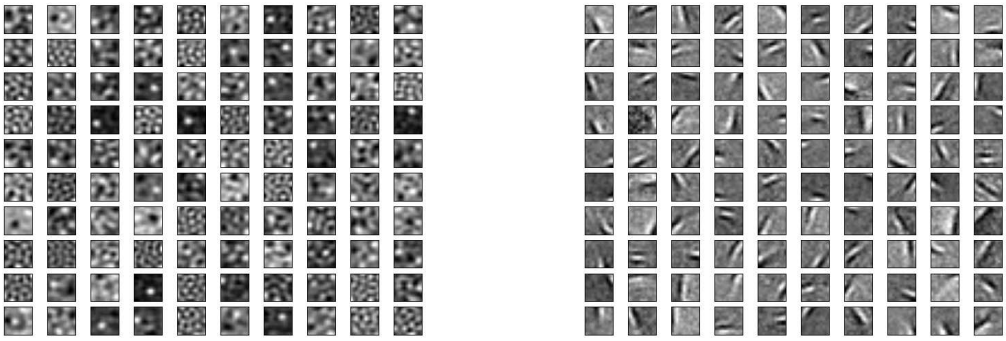


Figure 6: Weight decay vs. Gaussian noise. We show typical filters learnt from natural image patches in the over-complete case (200 hidden units). *Left:* regular autoencoder with weight decay. We tried a wide range of weight-decay values and learning rates: filters never appeared to capture a more interesting structure than what is shown here. Note that some local blob detectors are recovered compared to using no weight decay at all (Figure 5 right). *Right:* a denoising autoencoder with additive Gaussian noise ($\sigma = 0.5$) learns Gabor-like local oriented edge detectors. Clearly the filters learnt are qualitatively very different in the two cases.

yielded a mixture of edge detectors and grating filters. Clearly different corruption types and levels can yield qualitatively different filters. But it is interesting to note that all three noise types we experimented with were able to yield some potentially useful edge detectors.

5.2 Feature Detectors Learnt from Handwritten Digits

We also trained denoising autoencoders on the 28×28 gray-scale images of handwritten digits from the MNIST data set. For this experiment, we used denoising autoencoders with tied weights, cross-entropy reconstruction error, and zero-masking noise. The goal was to better understand the qualitative effect of the noise level. So we trained several denoising autoencoders, all starting from *the same initial random point in weight space*, but *with different noise levels*. Figure 8 shows some of the resulting filters learnt and how they are affected as we increase the level of corruption. With 0% corruption, the majority of the filters appear totally random, with only a few that specialize as little ink blob detectors. With increased noise levels, a much larger proportion of interesting (visibly non random and with a clear structure) feature detectors are learnt. These include local oriented stroke detectors and detectors of digit parts such as loops. It was to be expected that denoising a more corrupted input requires detecting bigger, less local structures: the denoising auto-encoder must rely on longer range statistical dependencies and pool evidence from a larger subset of pixels. Interestingly, filters that start from the same initial random weight vector often look like they "grow" from random, to local blob detector, to slightly bigger structure detectors such as a stroke detector, as we use increased noise levels. By "grow" we mean that the slightly larger structure learnt at a higher noise level often appears related to the smaller structure obtained at lower noise levels, in that they share about the same position and orientation.

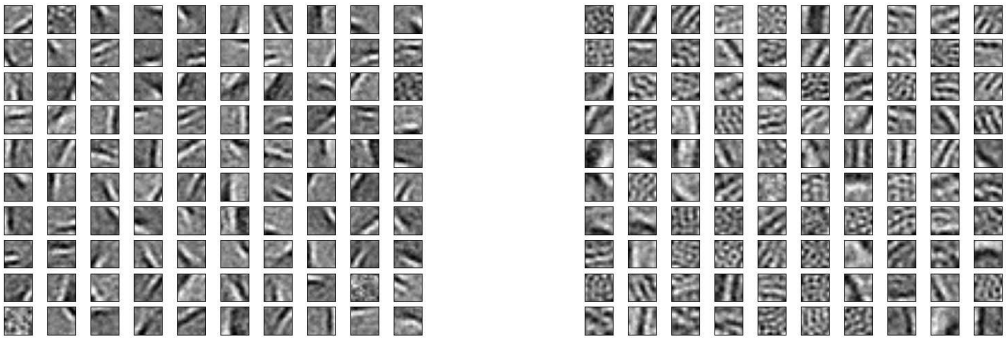


Figure 7: Filters obtained on natural image patches by denoising autoencoders using other noise types. *Left:* with 10% salt-and-pepper noise, we obtain oriented Gabor-like filters. They appear slightly less localized than when using Gaussian noise (contrast with Figure 6 right). *Right:* with 55% zero-masking noise we obtain filters that look like oriented gratings. For the three considered noise types, denoising training appears to learn filters that capture meaningful natural image statistics structure.

6. Experiments on Stacked Denoising Autoencoders

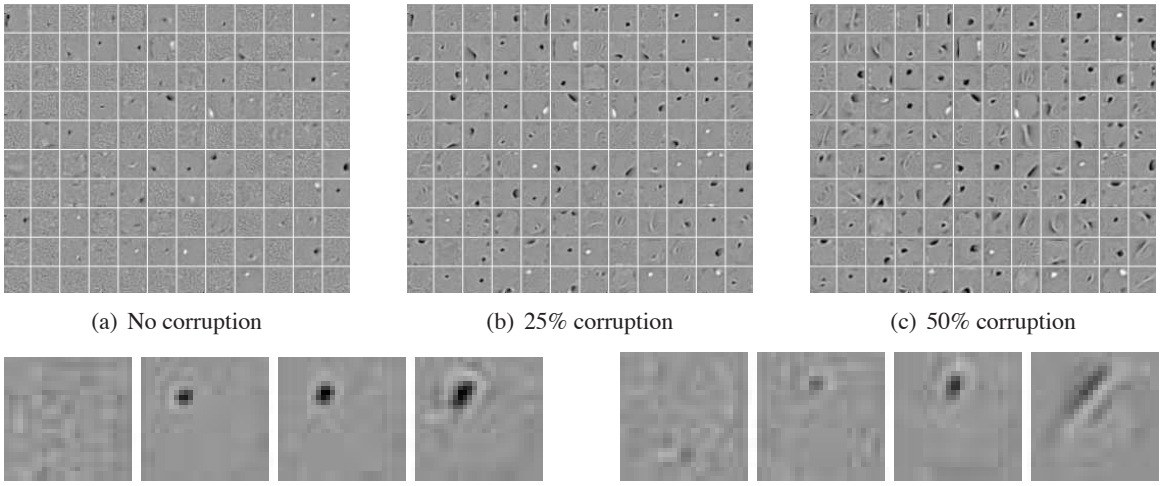
In this section, we evaluate denoising autoencoders as a pretraining strategy for building deep networks, using the stacking procedure that we described in Section 3.5. We shall mainly compare the classification performance of networks pretrained by stacking denoising autoencoders (SDAE), versus stacking regular autoencoders (SAE), versus stacking restricted Boltzmann machines (DBN), on a benchmark of classification problems.

6.1 Considered Classification Problems and Experimental Methodology

We considered 10 classification problems, the details of which are listed in Table 1. They consist of:

- The standard MNIST digit classification problem with 60000 training examples.
- The eight benchmark image classification problems used in Larochelle et al. (2007) which include more challenging variations of the MNIST digit classification problem (all with 10000 training examples), as well as three artificial 28×28 binary image classification tasks.[11] These problems were designed to be particularly challenging to current generic learning algorithms (Larochelle et al., 2007). They are illustrated in Figure 9.
- A variation of the *tzanetakis* audio genre classification data set (Bergstra, 2006) which contains 10000 three-second audio clips, equally distributed among 10 musical genres: blues, classical, country, disco, hiphop, pop, jazz, metal, reggae and rock. Each example in the set

11. The data sets for this benchmark are available at `http://www.iro.umontreal.ca/~lisa/icml2007`.



(a) No corruption (b) 25% corruption (c) 50% corruption

(d) Neuron A (0%, 10%, 20%, 50% corruption) (e) Neuron B (0%, 10%, 20%, 50% corruption)

Figure 8: Filters learnt by denoising autoencoder on MNIST digits, using zero-masking noise. *(a-c)* show some of the filters learnt by denoising autoencoders trained with various corruption levels ν. Filters at the same position in the three images are related only by the fact that the autoencoders were started from the same random initialization point in parameter space. *(d)* and *(e)* zoom in on the filters obtained for two of the neurons. As can be seen, with no noise, many filters remain similarly uninteresting (undistinctive almost uniform random grey patches). As we increase the noise level, denoising training forces the filters to differentiate more, and capture more distinctive features. Higher noise levels tend to induce less local filters, as expected. One can distinguish different kinds of filters, from local blob detectors, to stroke detectors, and character parts detectors at the higher noise levels.

was represented by 592 *Mel Phon Coefficient* (MPC) features. These are a simplified formulation of the *Mel-frequency cepstral coefficients* (MFCCs) that were shown to yield better classification performance (Bergstra, 2006).

All problems except *tzanetakis* had their data split into training set, validation set and test set. We kept the same standard splits that were used in Larochelle et al. (2007). The training set is used for both pretraining and fine tuning of the models. Classification performance on the validation set is used for choosing the best configuration of hyperparameters (model selection). The corresponding classification performance on the test set is then reported together with a 95% confidence interval.

For *tzanetakis* we used a slightly different procedure, since there was no predefined standard split and fewer examples. We used 10-fold cross validation, where each fold consisted of 8000 training examples, 1000 test and 1000 validation examples. For each fold, hyperparameters were chosen based on the performance on the validation set, and the retained model was used for computing the corresponding test error. We report the average test error and standard deviation across the 10 folds.

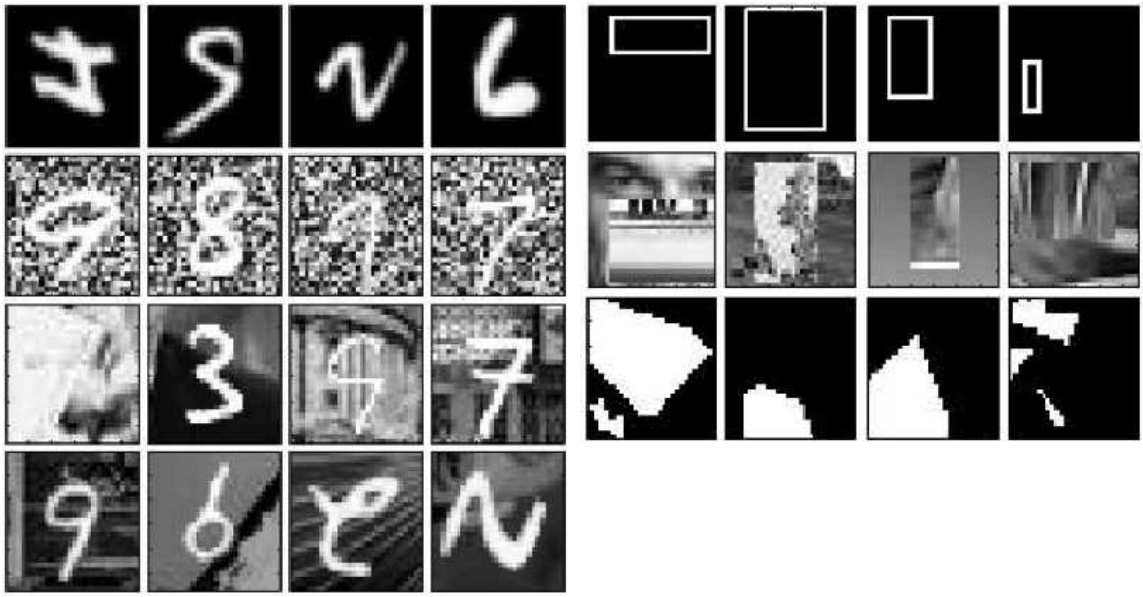
We were thus able to compare the classification performance of deep neural networks using different unsupervised initialization strategies for their parameters:

- MLP random: multilayer perceptron with usual random initialization;
- DBN (deep belief networks) corresponds to stacked RBM pretraining;
- SAE stacked autoencoder pretraining;
- SDAE stacked denoising autoencoder pretraining.

In all cases, the same supervised fine-tuning procedure was then used, that is, simple stochastic gradient descent with early stopping based on validation set performance.

Data Set	**Description**	**input**	m	**Train-Valid-Test**
MNIST	Standard MNIST digit classification problem.	784 gray-scale values scaled to [0,1]	10	50000-10000-10000
basic	Smaller subset of MNIST.		10	10000-2000-50000
rot	MNIST digits with added random rotation.		10	10000-2000-50000
bg-rand	MNIST digits with random noise background.		10	10000-2000-50000
bg-img	MNIST digits with random image background.		10	10000-2000-50000
bg-img-rot	MNIST digits with rotation and image background.		10	10000-2000-50000
rect	Discriminate between tall and wide rectangles (white on black).	784 values $\in\{0,1\}$	2	10000-2000-50000
rect-img	Discriminate between tall and wide rectangular image overlayed on a different background image.	784 values $\in [0,1]$	2	10000-2000-50000
convex	Discriminate between convex and concave shape.	784 values $\in\{0,1\}$	2	6000-2000-50000
tzanetakis	Classify genre of thirty second audio-clip.	592 MPC coefficients, standardized.	10	10-fold cross validation on 10000 training samples.

Table 1: Data sets. Characteristics of the 10 different problems considered. Except for *tzanetakis* whose input is made of 592 MPC features extracted from short audio sequences, all other problems are 28×28 gray-scale image classification tasks (i.e., input dimensionality is $28 \times 28 = 784$). See Larochelle et al. (2007) and Bergstra (2006) for further details on these data sets. The table gives, for each task, its shorthand name, a description of the problem, a description of input preprocessing, the number of classes (m), and the number of examples used for the training, validation and test sets respectively.



(a) *rot*, *bg-rand*, *bg-img*, *bg-img-rot*

(b) *rect*, *rect-img*, *convex*

Figure 9: Samples form the various image classification problems. (a): harder variations on the MNIST digit classification problems. (b): artificial binary classification problems.

On the 28×28 gray-scale image problems, SAE and SDAE used linear+sigmoid decoder layers and were trained using a cross-entropy loss, due to this being a natural choice for this kind of (near) binary images, as well as being functionally closer to RBM pretraining we wanted to compare against.

However for training the *first* layer on the *tzanetakis* problem, that is, for reconstructing *MPC coefficients*, a linear decoder and a squared reconstruction cost were deemed more appropriate (subsequent layers used sigmoid cross entropy as before). Similarly the first layer RBM used in pretraining a DBN on *tzanetakis* was defined with a Gaussian visible layer.

Table 2 lists the hyperparameters that had to be tuned by proper model selection (based on validation set performance). Note that to reduce the choice space, we restricted ourselves to the same number of hidden units, the same noise level, and the same learning rate for *all* hidden layers.

6.2 Empirical Comparison of Deep Network Training Strategies

Table 3 reports the classification performance obtained on all data sets using a 3 hidden layer neural network pretrained with the three different strategies: by stacking denoising autoencoders (SDAE-3), stacking restricted Boltzmann machines (DBN-3), and stacking regular autoencoders (SAE-3). For reference the table also contains the performance obtained by a single hidden-layer DBN-1 and by a Support Vector Machine with a RBF kernel (SVM_{rbf}).[12]

12. SVMs were trained using the libsvm implementation. Their hyperparameters (C and kernel width) were tuned semi-automatically (i.e., by human guided grid-search), searching for the best performer on the validation set.

hyperparameter	description	considered values
nHLay	number of hidden layers	{1,2,3}
nHUnit	number of units per hidden layer (same for all layers)	{1000,2000,3000}
lRate	fixed learning rate for unsupervised pretraining	$\{ 5 \times 10^{-6}, 5 \times 10^{-5}, 5 \times 10^{-4}, 5 \times 10^{-3}, 5 \times 10^{-2}, 10^{-1}\}$
lRateSup	fixed learning rate for supervised fine-tuning	{0.0005,0.005,0.05,0.1,0.15,0.2}
nEpoq	number of pretraining epochs (passages through the whole training set)	{5,10,50,100,125,150,200,300}
ν	corrupting noise level	fraction of corrupted inputs $(0, 0.10, 0.25, 0.40)$ or standard deviation for Gaussian noise $(0, 0.05, 0.10, 0.15, 0.30, 0.50)$

Table 2: List of hyperparameters for deep networks. These hyperparameters are common to all considered deep network training strategies, except for noise level ν which is specific to SDAE (for which we must also choose the kind of corruption). We list the typical values we considered in the majority of our experiments. Best performing configuration on the validation set was always searched for in a semi-automatic fashion, that is, running experiments in parallel on a large computation cluster, but with manual guidance to avoid wasting resources on unnecessary parts of the configuration space. Some experiments meant to study more closely the influence of specific hyperparameters occasionally used a finer search grid for them, as will be specified in the description of these experiments.

In these experiments, SDAE used a zero-masking corruption noise for all problems except for *tzanetakis*, for which a Gaussian noise was deemed more appropriate due to the nature of the input.

We see that SDAE-3 systematically outperforms the baseline SVM, as well as SAE-3 (except for *convex* for which the difference is not statistically significant). This shows clearly that denoising pretraining with a non-zero noise level is a better strategy than pretraining with regular autoencoders. For all but one problem, SDAE-3 is either the best performing algorithm or has its confidence interval overlap with that of the winning algorithm (i.e., difference cannot be considered statistically significant). In most cases, stacking 3 layers of denoising autoencoder seems to be on par or better than stacking 3 layers of RBMs in DBN-3.

In the following subsections, we will be conducting further detailed experiments to shed light on particular aspects of the denoising autoencoder pretraining strategy.

6.3 Influence of Number of Layers, Hidden Units per Layer, and Noise Level

Next we wanted to study more closely the influence of important architectural hyperparameters, namely the number of layers, the number of hidden units per layer, and the noise level. For this finer

Data Set	SVM_{rbf}	DBN-1	SAE-3	DBN-3	SDAE-3 (ν)
MNIST	**1.40**±0.23	**1.21**±0.21	**1.40**±0.23	**1.24**±0.22	**1.28**±0.22 (25%)
basic	**3.03**±0.15	3.94±0.17	3.46±0.16	3.11±0.15	**2.84**±0.15 (10%)
rot	11.11±0.28	14.69±0.31	10.30±0.27	10.30±0.27	**9.53**±0.26 (25%)
bg-rand	14.58±0.31	9.80±0.26	11.28±0.28	**6.73**±0.22	10.30±0.27 (40%)
bg-img	22.61±0.37	**16.15**±0.32	23.00±0.37	**16.31**±0.32	**16.68**±0.33 (25%)
bg-img-rot	55.18±0.44	52.21±0.44	51.93±0.44	47.39±0.44	**43.76**±0.43 (25%)
rect	**2.15**±0.13	4.71±0.19	2.41±0.13	2.60±0.14	**1.99**±0.12 (10%)
rect-img	24.04±0.37	23.69±0.37	24.05±0.37	22.50±0.37	**21.59**±0.36 (25%)
convex	19.13±0.34	19.92±0.35	**18.41**±0.34	**18.63**±0.34	**19.06**±0.34 (10%)
tzanetakis	**14.41**±2.18	18.07±1.31	**16.15**±1.95	18.38±1.64	**16.02**±1.04(0.05)

Table 3: Comparison of stacked denoising autoencoders (SDAE-3) with other models. Test error rate on all considered classification problems is reported together with a 95% confidence interval. Best performer is in bold, as well as those for which confidence intervals overlap. SDAE-3 appears to achieve performance superior or equivalent to the best other model on all problems except *bg-rand*. For SDAE-3, we also indicate the fraction ν of corrupted input components, or in case of *tzanetakis*, the standard deviation of the Gaussian noise, as chosen by proper model selection. Note that SAE-3 is equivalent to SDAE-3 with $\nu = 0\%$.

grained series of experiments, we chose to concentrate on the hardest of the considered problems, that is, the one with the most factors of variation: *bg-img-rot*.

We first examine how the proposed network training strategy behaves as we increase the capacity of the model both in breadth (number of neurons per layer) and in depth (number of hidden layers). Figure 10 shows the evolution of the performance as we increase the number of hidden layers from 1 to 3, for three different network training strategies: without any pretraining (standard MLP), with ordinary autoencoder pretraining (SAE) and with denoising autoencoder pretraining (SDAE). We clearly see a strict ordering: denoising pretraining being better than autoencoder pretraining being better than no pretraining. The advantage appears to increase with the number of layers (note that without pretraining it seems impossible to successfully train a 3 hidden layer network) and with the number of hidden units. This general behavior is a typical illustration of what is gained by pretraining deep networks with a good unsupervised criterion, and appears to be common to several pretraining strategies. We refer the reader to Erhan et al. (2010) for an empirical study and discussion regarding possible explanations for the phenomenon, centered on the observation of *regularization* effects (we exploit the hypothesis that features of X that help to capture $P(X)$ also help to capture $P(Y|X)$) and *optimization* effects (unsupervised pre-training initializes parameters near a better *local minimum* of *generalization* error).

Notice that in tuning the hyperparameters for all classification performances so far reported, we considered only a coarse choice of noise levels ν (namely 0%, 10%, 25%, or 40% of zero-masking corruption for the image classification problems). Clearly it was not necessary to pick the noise level very precisely to obtain good performances. In Figure 11 we examine in more details the influence of the level of corruption ν using a more fine-grained grid for problem *bg-img-rot*. We

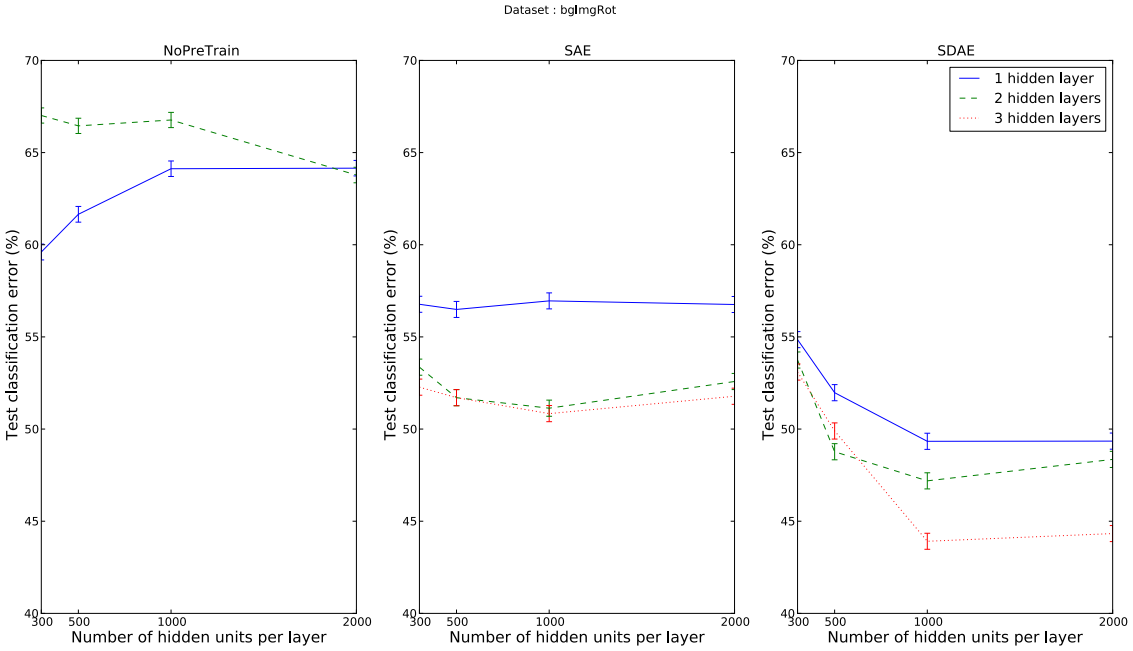


Figure 10: Classification performance on *bg-img-rot* for standard MLP with random initialization (NoPreTrain, left), SAE (middle), and SDAE (right), as we increase the number of hidden layers and the number of neurons per layer. Error bars show 95% confidence intervals. Note that without pretraining the curve for 3 layers is outside the graphic, the classification error being above 89%.

notice that SDAE appears to perform better than SAE (0 noise) for a rather wide range of noise levels, regardless of the number of hidden layers.

The following section reports an experiment that was conducted on three other data sets. The experiment had a different goal and thus used a coarser ν grid, but the resulting curves (see Figure 12) appear to follow a similar pattern to the one seen here (Figure 11).

6.4 Denoising Pretraining vs. Training with Noisy Input

We discussed in Section 4.2 the important distinction between denoising pretraining as it is done in SDAE and simply training with noisy inputs. SDAE uses a denoising criterion to learn good initial feature extractors at each layer that will be used as initialization for a *noiseless* supervised training. This is very different from training with noisy inputs, which amounts to training with a (virtually) expanded data set. This latter approach can in principle be applied to any classifier, such as an SVM[13] or a SAE. Note that in the case of the SAE, since there are two phases (pretraining and fine-tuning), it is possible to use noisy inputs for only the pretraining or for both the pretraining

13. For SAE, input examples can cheaply be corrupted on the fly, but this is not an option with standard SVM algorithms. So for SVM training we first augmented the training set by generating 9 extra variations of each original training example thus yielding a training set 10 times bigger than the original. Alternatively, we could instead have used the

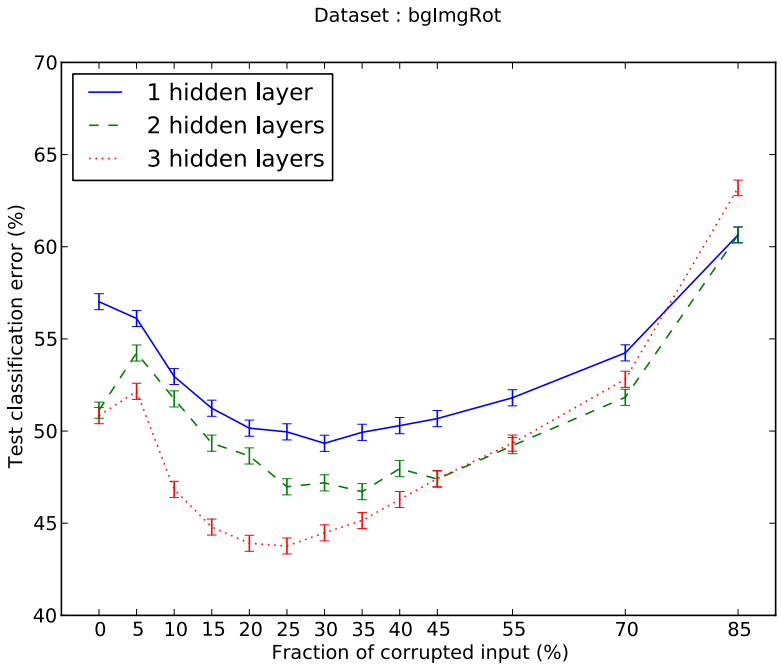


Figure 11: Sensitivity to the level of corruption. The curves report the test error rate for SDAE trained on problem *bg-img-rot* as a function of the fraction ν of corrupted input components (using zero masking noise). Error bars show 95% confidence interval. Note that 0% corruption corresponds to a SAE (regular autoencoder).

and fine-tuning phase. We experimentally compare these different approaches on three data sets in Figure 12. We see that denoising pretraining with SDAE, for a large range of noise levels, yields significantly improved performance, whereas training with noisy inputs sometimes degrades the performance, and sometimes improves it slightly but is clearly less beneficial than SDAE.

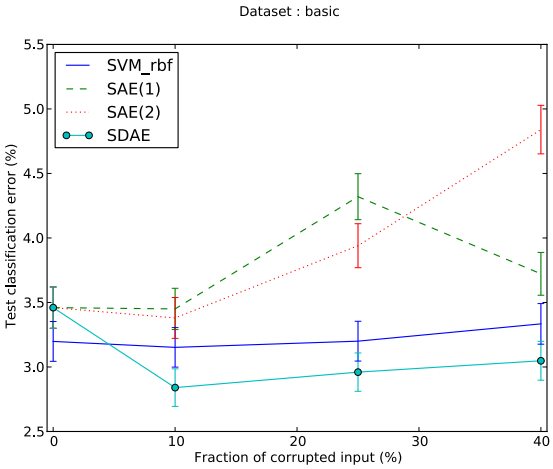
6.5 Variations on the Denoising Autoencoder: Alternate Corruption Types and Emphasizing

In the next series of experiments, we wanted to evaluate quantitatively the effect of using the various corrupting noises described in Section 3.3 as well as the *emphasized denoising autoencoder* variant described in Section 3.4.

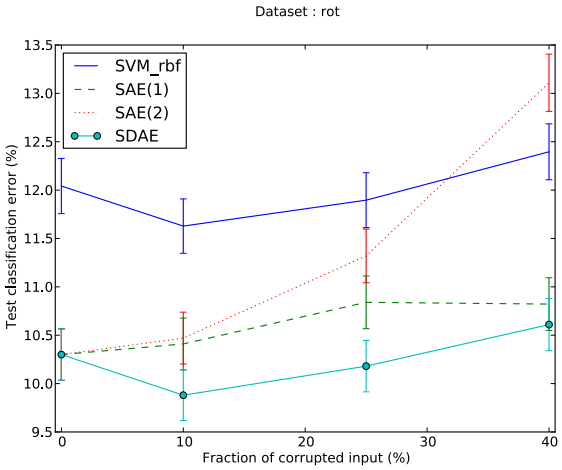
Extensive experiments were carried out on the same 3 problems we used in the previous section. Besides zero-masking noise (MN) we trained 3 hidden layer SDAE using salt-and-pepper noise (SP) and additive Gaussian noise (GS). For MN and SP, we also tried the emphasized variant.[14] For each considered variant, hyperparameters were selected as usual to yield the best performance on the

virtual SV technique (Schölkopf et al., 1996), which may or may not have yielded better performance, but since our main focus here is comparing noisy SAE with SDAE, SVM only serves as a simple baseline.

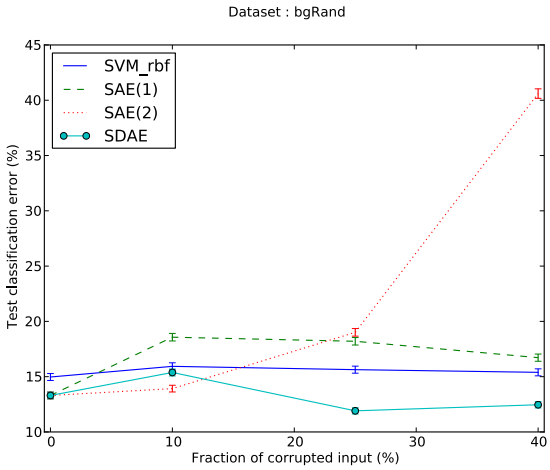
14. As already mentioned previously, since Gaussian noise corrupts every dimension, emphasized denoising does not make sense for this type of corruption.



(a) *basic*



(b) *rot*



(c) *bg-rand*

Figure 12: SDAE vs. training with noisy input. The test error of a **SDAE** with 3 hidden layers is compared to other algorithms trained with noisy inputs: a SVM with RBF kernel (**SVM$_{\text{rbf}}$**), a 3-hidden-layers SAE where noisy inputs were used for pretraining only (**SAE(1)**) and one where noisy inputs were used both for pretraining and supervised fine-tuning (**SAE(2)**). Hidden layers have 1000 neurons each. Zero-masking noise was used. Note that at 0% noise, the three stacked models correspond to an ordinary SAE. Error bars show 95% confidence interval. Denoising pretraining with SDAE appears to always yield significantly better performance, unlike training with noisy inputs.

Model	*basic*	*rot*	*bg-rand*
SVM$_{rbf}$	3.03±0.15	11.11±0.28	14.58±0.31
SAE-3	3.46±0.16	10.30±0.27	11.28±0.28
DBN-3	3.11±0.15	10.30±0.27	**6.73**±0.22
SDAE-3$_{MN}(\nu)$	2.84±0.15(10%)	9.53±0.26(25%)	10.30±0.27(40%)
SDAE-3$_{MN}(\nu)$ **+ emph**	**2.76**±0.14(25%)	10.36±0.27(25%)	9.69±0.26(40%)
SDAE-3$_{SP}(\nu)$	**2.66**±0.14(25%)	9.33±0.25(25%)	10.03±0.26(25%)
SDAE-3$_{SP}(\nu)$ **+ emph**	**2.48**±0.14(25%)	**8.76**±0.29(25%)	8.52±0.24(10%)
SDAE-3$_{GS}(\nu)$	**2.61**±0.14(0.1)	**8.86**±0.28(0.3)	11.73±0.28(0.1)

Table 4: Variations on 3-hidden-layer stacked denoising autoencoders (SDAE-3): alternative noise types and effect of emphasis. Considered noise types are masking noise (MN), salt-and-pepper (SP) and Gaussian noise (GS). Emphasized version considered double emphasis and full emphasis (see main text for detailed explanation). For easy comparison, the table also reproduces previously shown results for SVM_{rbf}, SAE-3, and DBN-3. Test error rate is reported together with a 95% confidence interval. Best performer is in bold, as well as those for which confidence intervals overlap. Corruption level ν (fraction of corrupted input components or Gaussian standard deviation) that was retained by model selection on the validation set is specified in parenthesis. SDAE-3$_{SP}$ with emphasis on reconstruction of corrupted dimension appears to be the best SDAE variant for these data sets, significantly improving performance on *rot* and *bg-rand*.

validation set. These included the number of units per layer (same for all layers), the corruption level ν (fraction of corrupted dimensions for MN and SP, or standard deviation for GS), with the usual considered values (listed previously in Table 2). For the emphasized version, a further hyperparameter was the degree of emphasis. We considered both *double emphasis*, where the weight on the reconstruction of the corrupted components is twice that on the uncorrupted components ($\alpha = 1,\ \beta = 0.5$), and *full emphasis* where all the weight is on reconstructing the corrupted components and none on the uncorrupted dimensions ($\alpha = 1,\ \beta = 0$). Table 4 reports the corresponding classification performance on the held-out test set. For the three considered data sets, an emphasized SDAE with salt-and-pepper noise appears to be the winning SDAE variant. It thus appears that a judicious choice of noise type and added emphasis may often buy us a better performance. However we had hoped, with these variants, to catch up with the performance of DBN-3 on the *bg-rand* problem,[15] but DBN-3 still performs significantly better than the best SDAE variant on this particular problem.

6.6 Are Features Learnt in an Unsupervised Fashion by SDAE Useful for SVMs?

In the following series of experiments, we wanted to verify whether the higher level representations extracted using SDAE could improve the performance of learning algorithms other than a neural network, such as SVMs.

15. As discussed in Larochelle et al. (2007), *bg-rand* is particularly favorable to RBMs because the pixel-wise independent noise perfectly matches what an RBM expects and will naturally not be represented in the hidden units.

To this end, we fed the representations learnt by the purely unsupervised phase of SDAE, at increasing higher levels (first, second and third hidden layer) to both a linear SVM and a Kernel SVM (using a RBF kernel). The hyperparameters of the SVM and its kernel were tuned on the validation set as usual. For computational reasons, we did not re-tune SDAE hyperparameters. Instead, we identified the best performing SDAE-pretrained neural networks with 1000 units per layer, based on their validation performance after fine-tuning from previous experiments, but used their saved weights prior to fine-tuning (i.e., after unsupervised denoising training only).

Results for all considered data sets are reported in Table 5, and Figure 13 highlights performance curves for two of them. Clearly, SVM performance can benefit significantly from using the higher level representation learnt by SDAE.[16] On all problems we see improved performance compared to using the original input (SVM_0). More interestingly, on most problems, SVM performance improves steadily as we use ever higher level representations. While it is not too surprising that linear SVMs can benefit from having the original input processed non-linearly, it is noteworthy that RBF kernel SVMs, which are high-capacity non-linear classifiers, also seem to benefit greatly from the non-linear mapping learned by SDAE.

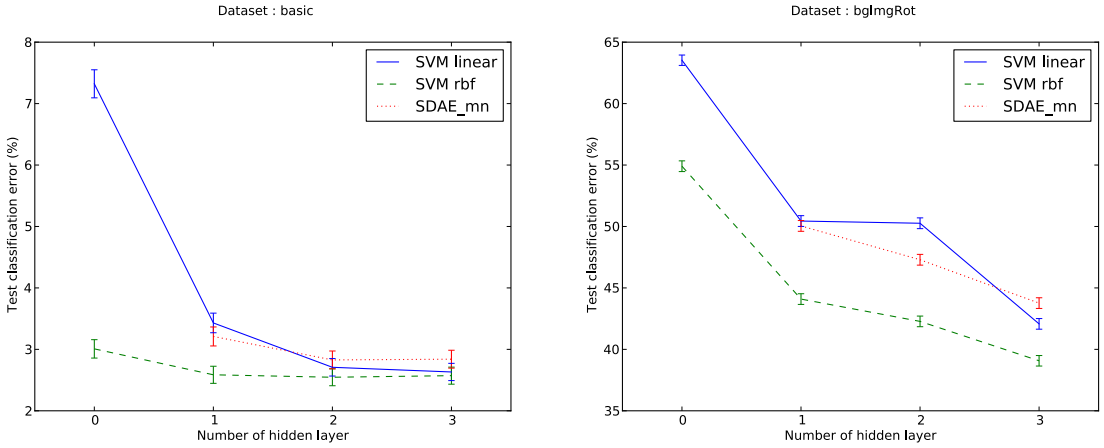


Figure 13: SVM on representations learnt by SDAE. The curves show evolution, on two data sets, of the test performance of linear and RBF kernel SVMs as we train them on higher level representations learnt in the unsupervised phase of SDAE. Performance of SDAE after supervised fine-tuning is also shown as $SDAE_{mn}$ (mn stands for masking noise). Hidden layer 0 corresponds to original input representation.

7. Generating Samples from Stacked Denoising Autoencoder Networks

Besides the classification performance comparisons and qualitative visual inspection of learnt filters, it is also customary in studies of deep generative models such as DBNs, to show samples generated

16. To verify that the learnt representation was responsible for the improved performance, rather than a random non-linear transformation, we also trained SVMs on the representation of the same neural network architecture but using randomly initialized weights: the performance degraded as we used the higher level representations.

Data Set	SVM kernel	SVM_0	SVM_1	SVM_2	SVM_3
MNIST	linear	5.33±0.44	1.49 ±0.24	**1.24**±0.22	**1.2**±0.21
	rbf	1.40±0.23	**1.04**±0.20	**0.94**±0.19	**0.95**±0.19
basic	linear	7.32±0.23	3.43±0.16	**2.71**±0.14	**2.63**±0.14
	rbf	3.03±0.15	**2.59**±0.14	**2.55**±0.14	**2.57**±0.14
rot	linear	43.47±0.43	21.74±0.36	15.15±0.31	10.00±0.26
	rbf	11.11±0.28	**8.45**±0.24	**8.27**±0.24	**8.64**±0.25
bg-rand	linear	24.14±0.38	13.58±0.30	13.00±0.29	11.32±0.28
	rbf	14.58±0.31	11.00±0.27	**10.08**±0.26	**10.16**±0.26
bg-img	linear	25.08±0.38	16.72±0.33	20.73±0.36	**14.55**±0.31
	rbf	22.61±0.37	15.91±0.32	16.36±0.32	**14.06**±0.30
bg-img-rot	linear	63.53±0.42	50.44±0.44	50.26±0.44	42.07±0.43
	rbf	55.18±0.44	44.09±0.44	42.28±0.43	**39.07**±0.43
rect	linear	29.04±0.40	6.43±0.22	2.31±0.13	1.80±0.12
	rbf	2.15±0.13	2.19±0.13	1.46±0.11	**1.22**±0.10
rect-img	linear	49.64±0.44	23.12±0.37	23.01±0.37	**21.43**±0.36
	rbf	24.04±0.37	22.27±0.36	21.56±0.36	**20.98**±0.36
convex	linear	45.75±0.44	24.10±0.37	18.40±0.34	**18.06**±0.34
	rbf	19.13±0.34	18.09±0.34	**17.39**±0.33	**17.53**±0.33
tzanetakis	linear	20.72±2.51	12.51±2.05	7.95±1.68	**5.04**±1.36
	rbf	14.41±2.18	7.54±1.64	**5.20**±1.38	**4.13**±1.23

Table 5: SVM performance on higher level representations learnt by SDAE. Performance of both linear SVM, and SVM with RBF kernel is reported, as they are trained on either original input (SVM_0), or on the representation learnt by a SDAE at the level of its first (SVM_1), second (SVM_2), or third (SVM_3) hidden layer. The representations used for the SVMs were those obtained prior to fine-tuning. Test error rate on all considered classification problems is reported together with a 95% confidence interval. Best performer is in bold, as well as those for which confidence intervals overlap. Clearly both linear and kernel SVM performance benefit from using the higher level representations learnt by SDAE. For most problems the performance increases steadily as we use representations from ever higher levels of the architecture.

from the trained models. This can yield another qualitative visual assessment of whether they were able to capture the input distribution well.

7.1 Top-Down Generation of a Visible Sample Given a Top-Layer Representation

Given a top-layer representation, a deep belief network (Hinton et al., 2006) is a directed graphical model, and it is easy to do a top down sampling pass, that is, sampling each layer conditioned on the layer above, to eventually produce a sample in the bottom layer that can be displayed. More precisely, in sigmoid deep belief networks (DBN), the representation at a lower layer X given the

layer above Y is distributed according to a product of independent Bernoullis whose mean is a deterministic function of Y, that is, $X|Y \sim \mathcal{B}(g_{\theta'}(Y))$, where $g_{\theta'}$ has the exact same form as that given in Equation 3 for the *decoder* part of an autoencoder. From a trained SAE or SDAE[17] it is thus possible to generate samples at one layer from the representation of the layer above in the exact same way as in a DBN.

7.2 Bottom-Up Inferring of the Top-Layer Representation Corresponding to a Given Input Pattern

In SAE/SDAE, *given an input representation at the bottom layer*, the corresponding representation in the top layer is computed in a deterministic *bottom-up* pass using encoding functions f_θ. The same procedure is used in DBNs and, in the graphical model perspective, can be viewed as an *approximate inference* of a factorial Bernoulli top-layer distribution given the low level input. This top-layer representation is to be understood as the parameters (the mean) of a factorial Bernoulli distribution for the actual binary units.

7.3 Generating Samples with SAE, SDAE, and DBN Using the Same Procedure

The deep belief network of Hinton et al. (2006) is a fully specified generative model. In particular the joint distribution of its top two layers is defined by an RBM model,[18], that is, an *undirected* graphical model from which one can efficiently sample using alternating Gibbs sampling (Hinton et al., 2006). So to sample from a DBN model, one would first sample from the top-layer RBM using alternating Gibbs sampling. Then, given the thus obtained top-layer representation, perform the single top down sampling pass previously described to produce a visible pattern at the bottom layer.

By contrast, SAE/SDAE training does not attempt to model the distribution of the top-layer representation. So even though—given a top-layer representation—we can use the exact same top down sampling procedure to generate input patterns from a SAE/SDAE as for a DBN, SAE/SDAE cannot by themselves alone be treated as fully specified generative models. They lack a model of the marginal distribution of their top layer.

We can easily fix this by modeling that top-layer distribution non-parametrically by the simple memory-based *empirical distribution* of the encoded top-layer representations of the n training set patterns. A visible sample can then be generated by simply taking the top-layer encoded representation of a randomly picked training set input, and carrying out the top-down sampling procedure explained previously, as illustrated in Figure 14. This same technique can also be used as an alternative sample-generation procedure for DBNs built by stacking RBMs.

If we keep the same fixed input pattern, and hence the same corresponding higher level representation, and perform several top-down samplings, we can thus observe what kind of pattern variations the deep multilayer part of a deep network has learnt to model (or abstract away in extracting the top-layer representation). Figure 15 shows the resulting variability in the regenerated patterns, for models pretrained respectively as SAE, SDAE[19] and DBN on MNIST without any su-

17. SAE and SDAE train such $g_{\theta'}$ to perform reconstruction, that is, predicting the mean value of a layer given the representation in the layer immediately above it.
18. This RBM was trained, using the training set, to model the representations obtained at the layer just below the top one, produced by the bottom-up pass we just described.
19. Both were pretrained with salt-and-pepper noise.

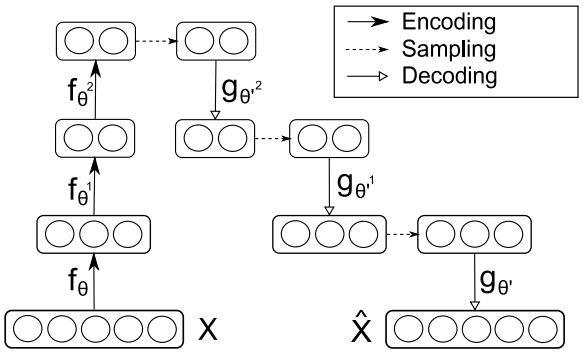


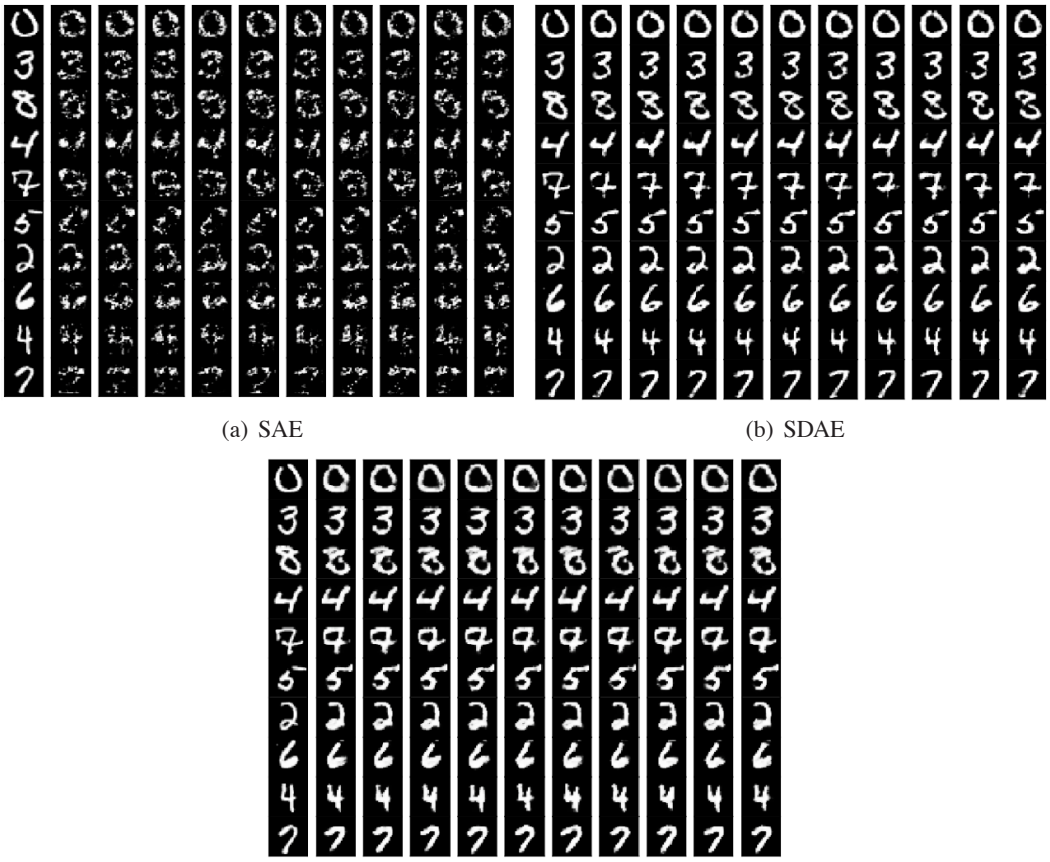
Figure 14: Non-parametric sampling procedure for pretrained networks. A randomly picked input form the original data set is provided as input. Its top level representation is obtained by deterministic bottom-up encoding using functions $f_{\theta^{(k)}}$. A visible pattern is generated given this high level representation, by alternating Bernoulli sampling and deterministic decoding, that is, by successively sampling from $\mathcal{B}(g_{\theta'^{(k)}}(\text{previouslayer}))$. This same procedure can be applied with SAE, SDAE and DBN. It allows to see the quality and variability of patterns one obtains given a high-level representation.

pervised fine-tuning. It appears that SDAE and DBN are able to resynthesize a variety of similarly good quality digits, whereas the SAE trained model regenerates patterns with much visible degradation in quality. This is further evidence of the qualitative difference resulting from optimizing a denoising criterion instead of mere reconstruction criterion. Note how SDAE puts back the missing hole in the loop of the regenerated 6, and sometimes straightens up the upper stroke of the 7, suggesting that it did indeed capture interesting specific characteristics. It appears that, when using this same sample generation procedure, SDAE and DBN yield a similar degree of variability in the regenerated patterns (with DBN patterns looking slightly fatter and SDAE patterns slightly thinner). Neither DBN nor SDAE guarantee that class boundaries will not be crossed,[20] for example DBN closes a loop in a 7 making it look closer to a 9, whereas SDAE sometimes breaks open the loop of an 8 making it look like a 3. But in all cases, and contrary to SAE, the regenerated patterns look like they could be samples from the same unknown input distribution that yielded the training set.

8. Conclusion and Future Work

The present work was inspired by recent successful approaches to training deep networks, specifically by their use of a local unsupervised criterion, and led by the question of what that criterion should be. At the same time we were motivated by a desire to bridge a remaining performance gap between deep belief networks and the stacking of ordinary autoencoders (Bengio et al., 2007; Larochelle et al., 2009a). This led us to address a theoretical shortcoming of traditional autoencoders—namely their inability in principle to learn useful over-complete representations—in a simple yet original way: by changing the objective from one involving mere reconstruction to the more challenging task of *denoising*. The resulting Stacked Denoising Autoencoder algorithm

20. The reader should however keep in mind that this results from unsupervised training only.



(a) SAE (b) SDAE

(c) DBN

Figure 15: Variability of the samples generated with 3-hidden-layer SAE, SDAE and DBN pre-trained models. Each sub-figure is to be read row-wise: the leftmost pattern in each row is a training set pattern. Following the sample generation depicted in Figure 14, it was provided as input to the network and its top-layer representation was computed by deterministic bottom up encoding. Patterns to its right were then generated independently given that top level representation. Clearly, SDAE trained networks, like DBNs, are able to regenerate high quality samples from their high level representation, contrary to SAE. SDAE and DBNs also appear to give rise to a similar level of variability in the bottom-up generated patterns (DBN patterns tending to be somewhat fatter). Note how SDAE puts back the missing hole in the loop of the regenerated 6, and sometimes straightens up the upper stroke of the last 7, suggesting that it did indeed capture meaningful specific characteristics. DBN and SDAE generated patterns can easily pass for samples from the unknown input distribution being modeled, unlike patterns generated by SAE.

for training deep networks, proved indeed able to bridge the performance gap with DBNs, yielding equivalent or better classification performance on all but one of the considered benchmark problems. As a deep network pretraining strategy, stacking of denoising autoencoders yielded in most cases a significant improvement in performance over the stacking of ordinary autoencoders. The representations thus extracted layer by layer, using a purely unsupervised local denoising criterion, appear to make subsequent classification tasks much easier. This is further evidenced by the fact that state-of-the-art shallow classifiers such as kernel SVMs also appear able to greatly benefit from it. Close examination of the feature extractors learnt by denoising autoencoders showed that they were able to zero in on useful structure in the data (such as Gabor-like edge detectors on natural image patches) that regular autoencoders seemed unable to learn.

The algorithm we developed is a straightforward, easy to implement, variation on the well-understood ordinary autoencoders. All that remains to be chosen is the kind and level of corrupting noise. It is likely that a careful choice, possibly guided by prior domain knowledge, may further boost application-specific performance. Nevertheless our experiments showed that high performance can already be achieved using very simple and generic noise types and with little tuning of the noise level. In addition, we were able to show that, contrary to what it may seem on the surface based on popular myths, the denoising training we advocate is *not* equivalent to using a mere weight decay regularization, nor is it the same as direct supervised training with corrupted (jittered) examples.

Beyond the specificities and practical usefulness of the simple algorithm we developed, **our results clearly establish the value of using a *denoising criterion* as an unsupervised objective to guide the learning of useful higher level representations.** This is in our view the most important contribution of our work, as it offers an interesting alternative to more usual (and often intractable) likelihood derived criteria. Indeed, denoising performance can easily be measured and directly optimized. The use of a denoising criterion is very different from the contrastive divergence training of RBMs or the direct enforcing of sparsity in autoencoders. We hope that our very encouraging results will inspire further research in this direction, both theoretical (to better understand the relationship between denoising and representation learning), and practical (to develop better learning algorithms based on this understanding).

There are certainly better ways to use denoising-based training signals in the learning of a deep network than the simple local approach we explored here. In particular, while stacking denoising autoencoders allows us to build a deep network, the denoising autoencoders we used here were shallow. It would thus be interesting to investigate deep denoising autoencoders with several hidden layers, and their ability to form useful representations. The choice and role of the corruption process also deserves further inquiry. If more involved corruption processes than those explored here prove beneficial, it would be most useful if they could be parameterized and learnt directly from the data, rather than having to be hand-engineered based on prior-knowledge.

Acknowledgments

This research was supported by funding from NSERC, MITACS, FQRNT, CIFAR, and the Canada Research Chairs, and partly carried out on computation resources made available by RQCHP.

References

G. An. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, 8(3):643–674, 1996.

H. Baird. Document image defect models. In *IAPR Workshop on Syntactic and Structural Pattern Recognition*, pages 38–46, Murray Hill, NJ., 1990.

P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

A. Bell and T.J. Sejnowski. The independent components of natural scenes are edge filters. *Vision Research*, 37:3327–3338, 1997.

A.J. Bell and T.J. Sejnowski. An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1): 1–127, 2009. Also published as a book. Now Publishers, 2009.

Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, June 2009.

Y. Bengio and Y. LeCun. Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press, 2007.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In Bernhard Schölkopf, John Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 153–160. MIT Press, 2007.

J. Bergstra. Algorithms for classifying recorded music by genre. Master's thesis, Université de Montreal, 2006.

J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.

C.M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7 (1):108–116, 1995.

H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

Y. Cho and L. Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, C. Williams, J. Lafferty, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS'09)*, pages 342–350. NIPS Foundation, 2010.

D. Erhan, Y. Bengio, A. Courville, P.A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, February 2010.

P. Gallinari, Y. LeCun, S. Thiria, and F. Fogelman-Soulie. Memoires associatives distribuees. In *Proceedings of COGNITIVA 87*, Paris, La Villette, 1987.

Y. Grandvalet, S. Canu, and S. Boucheron. Noise injection: Theoretical prospects. *Neural Computation*, 9(5):1093–1108, 1997.

J. Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th annual ACM Symposium on Theory of Computing*, pages 6–20, Berkeley, California, 1986. ACM Press.

J. Håstad and M. Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1:113–129, 1991.

D. Heckerman, D.M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

G.E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.

G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.

G.E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

L. Holmstrm and P. Koistinen. Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks*, 3(1):24–38, 1992.

J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, 79, 1982.

D.H. Hubel and T.N. Wiesel. Receptive fields of single neurons in the cat's striate cortex. *Journal of Physiology*, 148:574–591, 1959.

V. Jain and S.H. Seung. Natural image denoising with convolutional networks. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Leon Bottou, editors, *Advances in Neural Information Processing Systems 21 (NIPS'08)*, 2008.

N. Japkowicz, S.J. Hanson, and M.A. Gluck. Nonlinear autoassociation is not equivalent to PCA. *Neural Computation*, 12(3):531–545, 2000.

H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In Z. Ghahramani, editor, *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML'07)*, pages 473–480. ACM, 2007.

H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10:1–40, January 2009a.

H. Larochelle, D. Erhan, and P. Vincent. Deep learning using robust interdependent codes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, pages 312–319, April 2009b.

Y. LeCun. *Modèles connexionistes de l'apprentissage*. PhD thesis, Université de Paris VI, 1987.

Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

H. Lee, C. Ekanadham, and A. Ng. Sparse deep belief net model for visual area V2. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS'07)*, pages 873–880, Cambridge, MA, 2008. MIT Press.

R. Linsker. An application of the principle of maximum information preservation to linear systems. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 1 (NIPS'88)*. Morgan Kaufmann, 1989.

J.L. McClelland, D.E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2. MIT Press, Cambridge, 1986.

B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

B.A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, 37:3311–3325, December 1997.

T. Poggio and T. Vetter. Recognition and structure from one 2d model view: Observations on prototypes, object classes and symmetries. Technical Report A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

M. Ranzato, C.S. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 1137–1144. MIT Press, 2007.

M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS'07)*, pages 1185–1192, Cambridge, MA, 2008. MIT Press.

R. Scalettar and A. Zee. Emergence of grandmother memory in feed forward networks: Learning with noise and forgetfulness. In D. Waltz and J. A. Feldman, editors, *Connectionist Models and Their Implications: Readings from Cognitive Science*, pages 309–332. Ablex, Norwood, 1988.

B. Schölkopf, C.J.C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In C. von der Malsburg, W. von Seelen, J. C. Vorbrggen, and B. Sendhoff, editors, *Lecture Notes in Computer Science (Vol 112), Artificial Neural Netowrks ICANN'96*, pages 47–52. Springer, 1996.

S.H. Seung. Learning continuous attractors in recurrent networks. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems 10 (NIPS'97)*, pages 654–660. MIT Press, 1998.

J. Sietsma and R. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4(1): 67–79, 1991.

P. Simard, B. Victorri, Y. LeCun, and J. Denker. Tangent prop - A formalism for specifying selected invariances in an adaptive network. In J.E. Moody S.J. Hanson and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4 (NIPS'91)*, pages 895–903, San Mateo, CA, 1992. Morgan Kaufmann.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge, 1986.

P.E. Utgoff and D.J. Stracuzzi. Many-layered learning. *Neural Computation*, 14:2497–2539, 2002.

P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol. Extracting and composing robust features with denoising autoencoders. In W.W. Cohen, A. McCallum, and S.T. Roweis, editors, *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, pages 1096–1103. ACM, 2008.

A. von Lehman, E.G. Paek, P.F. Liao, A. Marrakchi, and J.S. Patel. Factors influencing learning by back-propagation. In *IEEE International Conference on Neural Networks*, volume 1, pages 335–341, San Diego 1988, 1988. IEEE, New York.

J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, pages 1168–1175, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390303.

Journal of Machine Learning Research 11 (2010) 3409-3451 Submitted 12/09; Revised 8/10; Published 12/10

L_p-Nested Symmetric Distributions

Fabian Sinz FABEE@TUEBINGEN.MPG.DE
Matthias Bethge MBETHGE@TUEBINGEN.MPG.DE
Werner Reichardt Center for Integrative Neuroscience
Bernstein Center for Computational Neuroscience
Max Planck Institute for Biological Cybernetics
Spemannstraße 41, 72076 Tübingen, Germany

Editor: Aapo Hyvärinen

Abstract

In this paper, we introduce a new family of probability densities called L_p-nested symmetric distributions. The common property, shared by all members of the new class, is the same functional form $\rho(\boldsymbol{x}) = \tilde{\rho}(f(\boldsymbol{x}))$, where f is a nested cascade of L_p-norms $\|\boldsymbol{x}\|_p = (\sum |x_i|^p)^{1/p}$. L_p-nested symmetric distributions thereby are a special case of ν-spherical distributions for which f is only required to be positively homogeneous of degree one. While both, ν-spherical and L_p-nested symmetric distributions, contain many widely used families of probability models such as the Gaussian, spherically and elliptically symmetric distributions, L_p-spherically symmetric distributions, and certain types of independent component analysis (ICA) and independent subspace analysis (ISA) models, ν-spherical distributions are usually computationally intractable. Here we demonstrate that L_p-nested symmetric distributions are still computationally feasible by deriving an analytic expression for its normalization constant, gradients for maximum likelihood estimation, analytic expressions for certain types of marginals, as well as an exact and efficient sampling algorithm. We discuss the tight links of L_p-nested symmetric distributions to well known machine learning methods such as ICA, ISA and mixed norm regularizers, and introduce the nested radial factorization algorithm (NRF), which is a form of non-linear ICA that transforms any linearly mixed, non-factorial L_p-nested symmetric source into statistically independent signals. As a corollary, we also introduce the uniform distribution on the L_p-nested unit sphere.

Keywords: parametric density model, symmetric distribution, ν-spherical distributions, non-linear independent component analysis, independent subspace analysis, robust Bayesian inference, mixed norm density model, uniform distributions on mixed norm spheres, nested radial factorization

1. Introduction

High-dimensional data analysis virtually always starts with the measurement of first and second-order moments that are sufficient to fit a multivariate Gaussian distribution, the maximum entropy distribution under these constraints. Natural data, however, often exhibit significant deviations from a Gaussian distribution. In order to model these higher-order correlations, it is necessary to have more flexible distributions available. Therefore, it is an important challenge to find generalizations of the Gaussian distribution which are more flexible but still computationally and analytically tractable. In particular, density models with an explicit normalization constant are desirable because they make direct model comparison possible by comparing the likelihood of held out test

samples for different models. Additionally, such models often allow for a direct optimization of the likelihood.

One way of imposing structure on probability distributions is to fix the general form of the iso-density contour lines. This approach was taken by Fernandez et al. (1995). They modeled the contour lines by the level sets of a positively homogeneous function of degree one, that is functions ν that fulfill $\nu(a \cdot \boldsymbol{x}) = a \cdot \nu(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^n$ and $a \in \mathbb{R}_0^+$. The resulting class of ν-spherical distributions have the general form $\rho(\boldsymbol{x}) = \tilde{\rho}(\nu(\boldsymbol{x}))$ for an appropriate $\tilde{\rho}$ which causes $\rho(\boldsymbol{x})$ to integrate to one. Since the only access of ρ to $\boldsymbol{x}$ is via ν one can show that, for a fixed ν, those distributions are generated by a univariate radial distribution. In other words, ν-spherically distributed random variables can be represented as a product of two independent random variables: one positive radial variable and another variable which is uniform on the 1-level set of ν. This property makes this class of distributions easy to fit to data since the maximum likelihood procedure can be carried out on the univariate radial distribution instead of the joint density. Unfortunately, deriving the normalization constant for the joint distribution in the general case is intractable because it depends on the surface area of those level sets which can usually not be computed analytically.

Known tractable subclasses of ν-spherical distributions are the Gaussian, elliptically contoured, and L_p-spherical distributions. The Gaussian is a special case of elliptically contoured distributions. After centering and whitening $\boldsymbol{x} := C^{-1/2}(\boldsymbol{s} - E[\boldsymbol{s}])$ a Gaussian distribution is spherically symmetric and the squared L_2-norm $||\boldsymbol{x}||_2^2 = x_1^2 + \cdots + x_n^2$ of the samples follow a χ^2-distribution (that is, the radial distribution is a χ-distribution). Elliptically contoured distributions other than the Gaussian are obtained by using a radial distribution different from the χ-distribution (Kelker, 1970; Fang et al., 1990).

The extension from L_2- to L_p-spherically symmetric distributions is based on replacing the L_2-norm by the L_p-norm

$$\nu(\boldsymbol{x}) = \|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}, \; p > 0$$

in the density definition. That is, the density of L_p-spherically symmetric distributions can always be written in the form $\rho(\boldsymbol{x}) = \tilde{\rho}(||\boldsymbol{x}||_p)$. Those distributions have been studied by Osiewalski and Steel (1993) and Gupta and Song (1997). We will adopt the naming convention of Gupta and Song (1997) and call $\|\boldsymbol{x}\|_p$ an *L_p-norm* even though the triangle inequality only holds for $p \geq 1$. L_p-spherically symmetric distributions with $p \neq 2$ are no longer invariant with respect to rotations (transformations from $SO(n)$). Instead, they are only invariant under permutations of the coordinate axes. In some cases, it may not be too restrictive to assume permutation or even rotational symmetry for the data. In other cases, such symmetry assumptions might not be justified and cause the model to miss important regularities.

Here, we present a generalization of the class of L_p-spherically symmetric distributions within the class of ν-spherical distributions that makes weaker assumptions about the symmetries in the data but still is analytically tractable. Instead of using a single L_p-norm to define the contour of the density, we use a nested cascade of L_p-norms where an L_p-norm is computed over groups of L_p-norms over groups of L_p-norms ..., each of which having a possibly different p. Due to this nested structure we call this new class of distributions *L_p-nested symmetric distributions*. The nested combination of L_p-norms preserves positive homogeneity but does not require permutation invariance anymore. While L_p-nested symmetric distributions are still invariant under reflections of the coordinate axes, permutation symmetry only holds within the subspaces of the L_p-norms at the bottom of

the cascade. As demonstrated in Sinz et al. (2009b), one possible application domain of L_p-nested symmetric distributions is natural image patches. In the current paper, we would like to present a formal treatment of this class of distributions. Readers interested in the application of these distributions to natural images should refer to Sinz et al. (2009b).

We demonstrate below that the construction of the nested L_p-norm cascade still bears enough structure to compute the Jacobian of polar-like coordinates similar to those of Song and Gupta (1997), and Gupta and Song (1997). With this Jacobian at hand it is possible to compute the univariate radial distribution for an arbitrary L_p-nested symmetric density and to define the uniform distribution on the L_p-nested unit sphere $\mathbb{L}_\nu = \{\boldsymbol{x} \in \mathbb{R}^n | \nu(\boldsymbol{x}) = 1\}$. Furthermore, we compute the surface area of the L_p-nested unit sphere and, therefore, the general normalization constant for L_p-nested symmetric distributions. By deriving these general relations for the class of L_p-nested symmetric distributions we have determined a new class of tractable ν-spherical distributions which is so far the only one containing the Gaussian, elliptically contoured, and L_p-spherical distributions as special cases.

L_p-spherically symmetric distributions have been used in various contexts in statistics and machine learning. Many results carry over to L_p-nested symmetric distributions which allow a wider application range. Osiewalski and Steel (1993) showed that the posterior on the location of a L_p-spherically symmetric distributions together with an improper Jeffrey's prior on the scale does not depend on the particular type of L_p-spherically symmetric distribution used. Below, we show that this results carries over to L_p-nested symmetric distributions. This means that we can robustly determine the location parameter by Bayesian inference for a very large class of distributions.

A large class of machine learning algorithms can be written as an optimization problem on the sum of a regularizer and a loss function. For certain regularizers and loss functions, like the sparse L_1 regularizer and the mean squared loss, the optimization problem can be seen as the maximum a posteriori (MAP) estimate of a stochastic model in which the prior and the likelihood are the negative exponentiated regularizer and loss terms. Since $\rho(\boldsymbol{x}) \propto \exp(-||\boldsymbol{x}||_p^p)$ is an L_p-spherically symmetric model, regularizers which can be written in terms of a norm have a tight link to L_p-spherically symmetric distributions. In an analogous way, L_p-nested symmetric distributions exhibit a tight link to mixed-norm regularizers which have recently gained increasing interest in the machine learning community (see, e.g., Zhao et al., 2008; Yuan and Lin, 2006; Kowalski et al., 2008). L_p-nested symmetric distributions can be used for a Bayesian treatment of mixed-norm regularized algorithms. Furthermore, they can be used to understand the prior assumptions made by such regularizers. Below we discuss an implicit dependence assumption between the regularized variables that follows from the theory of L_p-nested symmetric distributions.

Finally, the only factorial L_p-spherically symmetric distribution (Sinz et al., 2009a), the p-generalized Normal distribution, has been used as an ICA model in which the marginals follow an exponential power distribution. This class of ICA is particularly suited for natural signals like images and sounds (Lee and Lewicki, 2000; Zhang et al., 2004; Lewicki, 2002). Interestingly, L_p-spherically symmetric distributions other than the p-generalized Normal give rise to a non-linear ICA algorithm called radial Gaussianization for $p = 2$ (Lyu and Simoncelli, 2009) or radial factorization for arbitrary p (Sinz and Bethge, 2009). As discussed below, L_p-nested symmetric distributions are a natural extension of the linear L_p-spherically symmetric ICA algorithm to ISA, and give rise to a more general non-linear ICA algorithm in the spirit of radial factorization.

The remaining part of the paper is structured as follows: in Section 2 we define polar-like coordinates for L_p-nested symmetrically distributed random variables and present an analytical expression

for the determinant of the Jacobian for this coordinate transformation. Using this expression, we define the uniform distribution on the L_p-nested unit sphere and the class of L_p-nested symmetric distributions for an arbitrary L_p-nested function in Section 3. In Section 4 we derive an analytical form of L_p-nested symmetric distributions when marginalizing out lower levels of the L_p-nested cascade and demonstrate that marginals of L_p-nested symmetric distributions are not necessarily L_p-nested symmetric. Additionally, we demonstrate that the only factorial L_p-nested symmetric distribution is necessarily L_p-spherically symmetric and discuss the implications of this result for mixed norm regularizers. In Section 5 we propose an algorithm for fitting arbitrary L_p-nested symmetric models. We derive a sampling scheme for arbitrary L_p-nested symmetric distributions in Section 6. In Section 7 we generalize a result by Osiewalski and Steel (1993) on robust Bayesian inference on the location parameter to L_p-nested symmetric distributions. In Section 8 we discuss the relationship of L_p-nested symmetric distributions to ICA and ISA, and their possible role as priors on hidden variables in over-complete linear models. Finally, we derive a non-linear ICA algorithm for linearly mixed non-factorial L_p-nested symmetric sources in Section 9 which we call nested radial factorization (NRF).

2. L_p-nested Functions, Coordinate Transformation and Jacobian

Consider the function

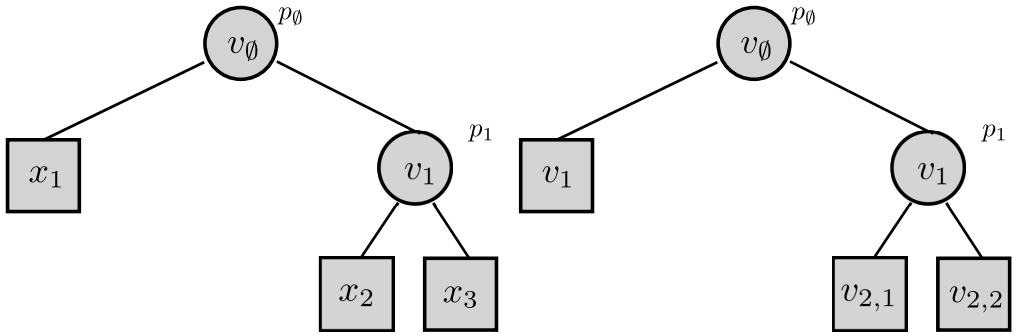
$$f(\boldsymbol{x}) = \left(|x_1|^{p_\emptyset} + \left(|x_2|^{p_1} + |x_3|^{p_1} \right)^{\frac{p_\emptyset}{p_1}} \right)^{\frac{1}{p_\emptyset}} \tag{1}$$

with $p_\emptyset, p_1 \in \mathbb{R}^+$. This function is obviously a cascade of two L_p-norms and is thus positively homogeneous of degree one. Figure 1(a) shows this function visualized as a tree. Naturally, any tree like the ones in Figure 1 corresponds to a function of the kind of Equation (1). In general, the n leaves of the tree correspond to the n coefficients of the vector $\boldsymbol{x} \in \mathbb{R}^n$ and each inner node computes the L_p-norm of its children using its specific p. We call the class of functions which is generated in this way *L_p-nested* and the corresponding distributions, which are symmetric or invariant with respect to it, *L_p-nested symmetric distributions*.

L_p-nested functions are much more flexible in creating different shapes of level sets than single L_p-norms. Those level sets become the iso-density contours in the family of L_p-nested symmetric distributions. Figure 2 shows a variety of contours generated by the simplest non-trivial L_p-nested function shown in Equation (1). The shapes show the unit spheres for all possible combinations of $p_\emptyset, p_1 \in \{0.5, 1, 2, 10\}$. On the diagonal, $p_\emptyset$ and p_1 are equal and therefore constitute L_p-norms. The corresponding distributions are members of the L_p-spherically symmetric class.

To make general statements about general L_p-nested functions, we introduce a notation that is suitable for the tree structure of L_p-nested functions. As we will heavily use that notation in the remainder of the paper, we would like to emphasize the importance of the following paragraphs. We will illustrate the notation with an example below. Additionally, Figure 1 and Table 1 can be used for reference.

We use multi-indices to denote the different nodes of the tree corresponding to an L_p-nested function f. The function $f = f_\emptyset$ itself computes the value $v_\emptyset$ at the root node (see Figure 1). Those values are denoted by variables v. The functions corresponding to its children are denoted by $f_1, ..., f_{\ell_\emptyset}$, that is, $f(\cdot) = f_\emptyset(\cdot) = \|(f_1(\cdot), ..., f_{\ell_\emptyset}(\cdot))\|_{p_\emptyset}$. We always use the letter "ℓ" indexed by the node's multi-index to denote the total number of direct children of that node. The functions of



(a) Equation (1) as tree. (b) Equation (1) as tree in multi-index notation.

Figure 1: Equation (1) visualized as a tree with two different naming conventions. Figure (a) shows the tree where the nodes are labeled with the coefficients of $\boldsymbol{x} \in \mathbb{R}^n$. Figure (b) shows the same tree in multi-index notation where the multi-index of a node describes the path from the root node to that node in the tree. The leaves $v_1, v_{2,1}$ and $v_{2,2}$ still correspond to x_1, x_2 and x_3, respectively, but have been renamed to the multi-index notation used in this article.

$f(\cdot) = f_\emptyset(\cdot)$	L_p-nested function
$I = i_1, ..., i_m$	Multi-index denoting a node in the tree: The single indices describe the path from the root node to the respective node I.
$\boldsymbol{x}_I$	All entries in $\boldsymbol{x}$ that correspond to the leaves in the subtree under the node I
$\boldsymbol{x}_{\widehat{I}}$	All entries in $\boldsymbol{x}$ that are not leaves in the subtree under the node I
$f_I(\cdot)$	L_p-nested function corresponding to the subtree under the node I
$v_\emptyset$	Function value at the root node
v_I	Function value at an arbitrary node with multi-index I
ℓ_I	The number of direct children of a node I
n_I	The number of leaves in the subtree under the node I
$\boldsymbol{v}_{I,1:\ell_I}$	Vector with the function values at the direct children of a node I

Table 1: Summary of the notation used for L_p-nested functions in this article.

the children of the i^{th} child of the root node are denoted by $f_{i,1}, ..., f_{i,\ell_i}$ and so on. In this manner, an index is added for denoting the children of a particular node in the tree and each multi-index denotes the path to the respective node in the tree. For the sake of compact notation, we use upper case letters to denote a single multi-index $I = i_1, ..., i_\ell$. The range of the single indices and the length of the multi-index should be clear from the context. A concatenation I, k of a multi-index I with a single index k corresponds to adding k to the index tuple, that is, $I, k = i_1, ..., i_m, k$. We use the

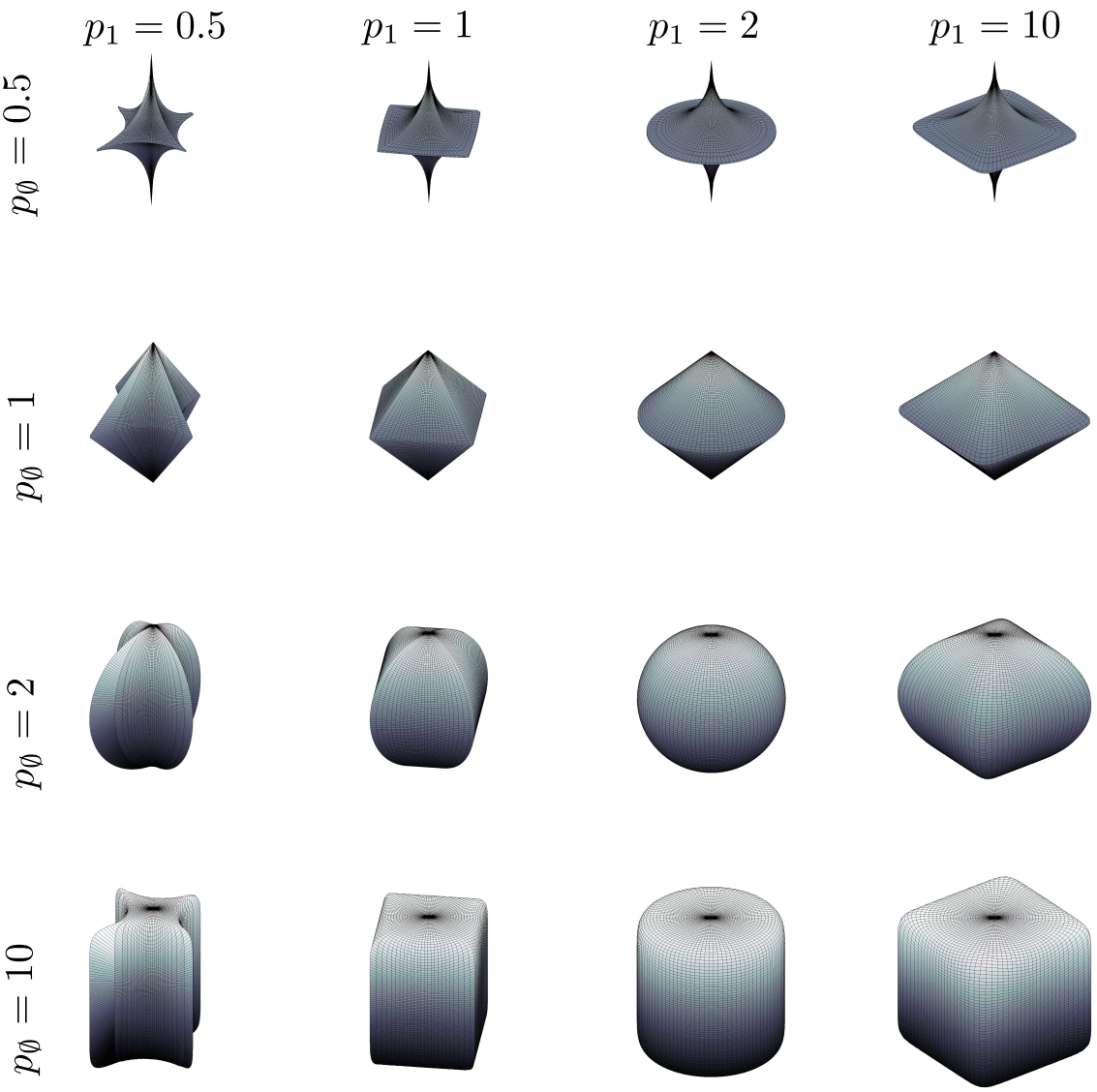


Figure 2: Variety of contours created by the L_p-nested function of Equation (1) for all combinations of $p_\emptyset, p_1 \in \{0.5, 1, 2, 10\}$.

convention that $I, \emptyset = I$. Those coefficients of the vector $\boldsymbol{x}$ that correspond to leaves of the subtree under a node with the index I are denoted by $\boldsymbol{x}_I$. The complement of those coefficients, that is, the ones that are not in the subtree under the node I, are denoted by $\boldsymbol{x}_{\hat{I}}$. The number of leaves in a subtree under a node I is denoted by n_I. If I denotes a leaf then $n_I = 1$.

The L_p-nested function associated with the subtree under a node I is denoted by

$$f_I(\boldsymbol{x}_I) = ||(f_{I,1}(\boldsymbol{x}_{I,1}), ..., f_{I,\ell_I}(\boldsymbol{x}_{I,\ell_I}))^\top||_{p_I}.$$

Just like for the root node, we use the variable v_I to denote the function value $v_I = f_I(\boldsymbol{x}_I)$ of a subtree I. A vector with the function values of the children of I is denoted with bold font $\boldsymbol{v}_{I,1:\ell_I}$ where the colon indicates that we mean the vector of the function values of the ℓ_I children of node I:

$$\begin{aligned} f_I(\boldsymbol{x}_I) &= ||(f_{I,1}(\boldsymbol{x}_{I,1}),...,f_{I,\ell_I}(\boldsymbol{x}_{I,\ell_I}))^\top||_{p_I} \\ &= ||(v_{I,1},...,v_{I,\ell_I})^\top||_{p_I} = ||\boldsymbol{v}_{I,1:\ell_I}||_{p_I}. \end{aligned}$$

Note that we can assign an arbitrary p to leaf nodes since ps for single variables always cancel. For that reason we can choose an arbitrary p for convenience and fix its value to $p = 1$. Figure 1(b) shows the multi-index notation for our example of Equation (1).

To illustrate the notation: Let $I = i_1,...,i_d$ be the multi-index of a node in the tree. $i_1,...,i_d$ describes the path to that node, that is, the respective node is the i_d^{th} child of the i_{d-1}^{th} child of the i_{d-2}^{th} child of the ... of the i_1^{th} child of the root node. Assume that the leaves in the subtree below the node I cover the vector entries $x_2,...,x_{10}$. Then $\boldsymbol{x}_I = (x_2,...,x_{10})$, $\boldsymbol{x}_{\widehat{I}} = (x_1,x_{11},x_{12},...)$, and $n_I = 9$. Assume that node I has $\ell_I = 2$ children. Those would be denoted by $I,1$ and $I,2$. The function realized by node I would be denoted by f_I and only acts on $\boldsymbol{x}_I$. The value of the function would be $f_I(\boldsymbol{x}_I) = v_I$ and the vector containing the values of the children of I would be $\boldsymbol{v}_{I,1:2} = (v_{I,1},v_{I,2})^\top = (f_{I,1}(\boldsymbol{x}_{I,1}),f_{I,2}(\boldsymbol{x}_{I,2}))^\top$.

We now introduce a coordinate representation specially tailored to L_p-nested symmetrically distributed variables: One of the most important consequences of the positive homogeneity of f is that it can be used to "normalize" vectors and, by that property, create a polar like coordinate representation of a vector $\boldsymbol{x}$. Such polar-like coordinates generalize the coordinate representation for L_p-norms by Gupta and Song (1997).

Definition 1 (Polar-like Coordinates) *We define the following polar-like coordinates for a vector* $\boldsymbol{x} \in \mathbb{R}^n$*:*

$$\begin{aligned} u_i &= \frac{x_i}{f(\boldsymbol{x})} \text{ for } i = 1,...,n-1, \\ r &= f(\boldsymbol{x}). \end{aligned}$$

The inverse coordinate transformation is given by

$$\begin{aligned} x_i &= r u_i \text{ for } i = 1,...,n-1, \\ x_n &= r \Delta_n u_n \end{aligned}$$

where $\Delta_n = \operatorname{sgn} x_n$ *and* $u_n = \frac{|x_n|}{f(\boldsymbol{x})}$.

Note that u_n is not part of the coordinate representation since normalization with $1/f(\boldsymbol{x})$ decreases the degrees of freedom $\boldsymbol{u}$ by one, that is, u_n can always be computed from all other u_i by solving $f(\boldsymbol{u}) = f(\boldsymbol{x}/f(\boldsymbol{x})) = 1$ for u_n. We use the term u_n only for notational simplicity. With a slight abuse of notation, we will use $\boldsymbol{u}$ to denote the normalized vector $\boldsymbol{x}/f(\boldsymbol{x})$ or only its first $n-1$ components. The exact meaning should always be clear from the context.

The definition of the coordinates is exactly the same as the one by Gupta and Song (1997) with the only difference that the L_p-norm is replaced by an L_p-nested function. Just as in the case of L_p-spherical coordinates, it will turn out that the determinant of the Jacobian of the coordinate

transformation does not depend on the value of Δ_n and can be computed analytically. The determinant is essential for deriving the uniform distribution on the unit L_p-nested sphere $\mathbb{L}_f$, that is, the 1-level set of f. Apart from that, it can be used to compute the radial distribution for a given L_p-nested symmetric distribution. We start by stating the general form of the determinant in terms of the partial derivatives $\frac{\partial u_n}{\partial u_k}$, u_k and r. Afterwards we demonstrate that those partial derivatives have a special form and that most of them cancel in Laplace's expansion of the determinant.

Lemma 2 (Determinant of the Jacobian) *Let r and $\boldsymbol{u}$ be defined as in Definition 1. The general form of the determinant of the Jacobian $\mathcal{J} = \left(\frac{\partial x_i}{\partial y_j}\right)_{ij}$ of the inverse coordinate transformation for $y_1 = r$ and $y_i = u_{i-1}$ for $i = 2,\ldots,n$, is given by*

$$|\det \mathcal{J}| = r^{n-1}\left(-\sum_{k=1}^{n-1}\frac{\partial u_n}{\partial u_k}\cdot u_k + u_n\right). \tag{2}$$

Proof The proof can be found in the Appendix A. ■

The problematic parts in Equation (2) are the terms $\frac{\partial u_n}{\partial u_k}$, which obviously involve extensive usage of the chain rule. Fortunately, most of them cancel when inserting them back into Equation (2), leaving a comparably simple formula. The remaining part of this section is devoted to computing those terms and demonstrating how they vanish in the formula for the determinant. Before we state the general case we would like to demonstrate the basic mechanism through a simple example. We urge the reader to follow this example as it illustrates all important ideas about the coordinate transformation and its Jacobian.

Example 1 *Consider an L_p-nested function very similar to our introductory example of Equation (1):*

$$f(\boldsymbol{x}) = \left(\left(|x_1|^{p_1} + |x_2|^{p_1}\right)^{\frac{p_\emptyset}{p_1}} + |x_3|^{p_\emptyset}\right)^{\frac{1}{p_\emptyset}}.$$

Setting $\boldsymbol{u} = \frac{\boldsymbol{x}}{f(\boldsymbol{x})}$ and solving for u_3 yields

$$f(\boldsymbol{u}) = 1 \Leftrightarrow\ u_3 = \left(1 - \left(|u_1|^{p_1} + |u_2|^{p_1}\right)^{\frac{p_\emptyset}{p_1}}\right)^{\frac{1}{p_\emptyset}}. \tag{3}$$

We would like to emphasize again, that u_3 is actually not part of the coordinate representation and only used for notational simplicity. By construction, u_3 is always positive. This is no restriction since Lemma 2 shows that the determinant of the Jacobian does not depend on its sign. However, when computing the volume and the surface area of the L_p-nested unit sphere, it will become important since it introduces a factor of 2 to account for the fact that u_3 (or u_n in general) can in principle also attain negative values.

Now, consider

$$G_2(\boldsymbol{u}_{\widehat{2}}) = g_2(\boldsymbol{u}_{\widehat{2}})^{1-p_\emptyset} = \left(1 - \left(|u_1|^{p_1} + |u_2|^{p_1}\right)^{\frac{p_\emptyset}{p_1}}\right)^{\frac{1-p_\emptyset}{p_\emptyset}},$$
$$F_1(\boldsymbol{u}_1) = f_1(\boldsymbol{u}_1)^{p_\emptyset - p_1} = \left(|u_1|^{p_1} + |u_2|^{p_1}\right)^{\frac{p_\emptyset - p_1}{p_1}},$$

where the subindices of $\boldsymbol{u}$, f, g, G *and* F *have to be read as multi-indices. The function* g_I *computes the value of the node* I *from all other leaves that are not part of the subtree under* I *by fixing the value of the root node to one.*

$G_2(\boldsymbol{u}_{\widehat{2}})$ *and* $F_1(\boldsymbol{u}_1)$ *are terms that arise from applying the chain rule when computing the partial derivatives* $\frac{\partial u_3}{\partial u_k}$. *Taking those partial derivatives can be thought of as peeling off layer by layer of Equation* (3) *via the chain rule. By doing so, we "move" on a path between* u_3 *and* u_k. *Each application of the chain rule corresponds to one step up or down in the tree. First, we move upwards in the tree, starting from* u_3. *This produces the G-terms. In this example, there is only one step upwards, but in general, there can be several, depending on the depth of* u_n *in the tree. Each step up will produce one G-term. At some point, we will move downwards in the tree to reach* u_k. *This will produce the F-terms. While there are as many G-terms as upward steps, there is one term less when moving downwards. Therefore, in this example, there is one term* $G_2(\boldsymbol{u}_{\widehat{2}})$ *which originates from using the chain rule upwards in the tree and one term* $F_1(\boldsymbol{u}_1)$ *from using it downwards. The indices correspond to the multi-indices of the respective nodes.*

Computing the derivative yields

$$\frac{\partial u_3}{\partial u_k} = -G_2(\boldsymbol{u}_{\widehat{2}})F_1(\boldsymbol{u}_1)\Delta_k|u_k|^{p_1-1}.$$

By inserting the results in Equation (2) we obtain

$$\begin{aligned}
\frac{1}{r^2}|\mathcal{J}| &= \sum_{k=1}^{2} G_2(\boldsymbol{u}_{\widehat{2}})F_1(\boldsymbol{u}_1)|u_k|^{p_1} + u_3 \\
&= G_2(\boldsymbol{u}_{\widehat{2}})\left(F_1(\boldsymbol{u}_1)\sum_{k=1}^{2}|u_k|^{p_1} + 1 - F_1(\boldsymbol{u}_1)F_1(\boldsymbol{u}_1)^{-1}\left(|u_1|^{p_1}+|u_2|^{p_1}\right)^{\frac{p_\emptyset}{p_1}}\right) \\
&= G_2(\boldsymbol{u}_{\widehat{2}})\left(F_1(\boldsymbol{u}_1)\sum_{k=1}^{2}|u_k|^{p_1} + 1 - F_1(\boldsymbol{u}_1)\sum_{k=1}^{2}|u_k|^{p_1}\right) \\
&= G_2(\boldsymbol{u}_{\widehat{2}}).
\end{aligned}$$

The example suggests that the terms from using the chain rule downwards in the tree cancel while the terms from using the chain rule upwards remain. The following proposition states that this is true in general.

Proposition 3 (Determinant of the Jacobian) *Let* $\mathcal{L}$ *be the set of multi-indices of the path from the leaf* u_n *to the root node (excluding the root node) and let the terms* $G_{I,\ell_I}(\boldsymbol{u}_{\widehat{I,\ell_I}})$ *recursively be defined as*

$$G_{I,\ell_I}(\boldsymbol{u}_{\widehat{I,\ell_I}}) = g_{I,\ell_I}(\boldsymbol{u}_{\widehat{I,\ell_I}})^{p_{I,\ell_I}-p_I} = \left(g_I(\boldsymbol{u}_{\widehat{I}})^{p_I} - \sum_{j=1}^{\ell-1} f_{I,j}(\boldsymbol{u}_{I,j})^{p_I}\right)^{\frac{p_{I,\ell_I}-p_I}{p_I}}$$

where each of the functions g_{I,ℓ_I} *computes the value of the* ℓ^{th} *child of a node* I *as a function of its neighbors* $(I,1)$, ..., (I,ℓ_I-1) *and its parent* I *while fixing the value of the root node to one. This is equivalent to computing the value of the node* I *from all coefficients* $\boldsymbol{u}_{\widehat{I}}$ *that are not leaves in the subtree under* I. *Then, the determinant of the Jacobian for an* L_p*-nested function is given by*

$$|\det \mathcal{J}| = r^{n-1}\prod_{L\in\mathcal{L}} G_L(\boldsymbol{u}_{\widehat{L}}).$$

Proof The proof can be found in the Appendix A. ■

Let us illustrate the determinant with two examples:

Example 2 *Consider a normal L_p-norm*

$$f(\boldsymbol{x}) = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

which is obviously also an L_p-nested function. Resolving the equation for the last coordinate of the normalized vector $\boldsymbol{u}$ yields $g_n(\boldsymbol{u}_{\widehat{n}}) = u_n = \left(1 - \sum_{i=1}^{n-1} |u_i|^p\right)^{\frac{1}{p}}$. Thus, the term $G_n(\boldsymbol{u}_{\widehat{n}})$ is given by $\left(1 - \sum_{i=1}^{n-1} |u_i|^p\right)^{\frac{1-p}{p}}$ which yields a determinant of $|\det \mathcal{J}| = r^{n-1} \left(1 - \sum_{i=1}^{n-1} |u_i|^p\right)^{\frac{1-p}{p}}$. This is exactly the one derived by Gupta and Song (1997).

Example 3 *Consider the introductory example*

$$f(\boldsymbol{x}) = \left(|x_1|^{p_\emptyset} + (|x_2|^{p_1} + |x_3|^{p_1})^{\frac{p_\emptyset}{p_1}} \right)^{\frac{1}{p_\emptyset}}.$$

Normalizing and resolving for the last coordinate yields

$$u_3 = \left((1 - |u_1|^{p_\emptyset})^{\frac{p_1}{p_\emptyset}} - |u_2|^{p_1} \right)^{\frac{1}{p_1}}$$

and the terms $G_2(\boldsymbol{u}_{\widehat{2}})$ and $G_{2,2}(\boldsymbol{u}_{\widehat{2,2}})$ of the determinant $|\det \mathcal{J}| = r^2 G_2(\boldsymbol{u}_{\widehat{2}}) G_{2,2}(\boldsymbol{u}_{\widehat{2,2}})$ are given by

$$G_2(\boldsymbol{u}_{\widehat{2}}) = (1 - |u_1|^{p_\emptyset})^{\frac{p_1 - p_\emptyset}{p_\emptyset}},$$
$$G_{2,2}(\boldsymbol{u}_{\widehat{2,2}}) = \left((1 - |u_1|^{p_\emptyset})^{\frac{p_1}{p_\emptyset}} - |u_2|^{p_1} \right)^{\frac{1-p_1}{p_1}}.$$

Note the difference to Example 1 where x_3 was at depth one in the tree while x_3 is at depth two in the current case. For that reason, the determinant of the Jacobian in Example 1 involved only one G-term while it has two G-terms here.

3. L_p-Nested Symmetric and L_p-Nested Uniform Distribution

In this section, we define the L_p-nested symmetric and the L_p-nested uniform distribution and derive their partition functions. In particular, we derive the surface area of an arbitrary L_p-nested unit sphere $\mathbb{L}_f = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) = 1\}$ corresponding to an L_p-nested function f. By Equation (5) of Fernandez et al. (1995) every ν-spherical and hence any L_p-nested symmetric density has the form

$$\rho(\boldsymbol{x}) = \frac{\phi(f(\boldsymbol{x}))}{f(\boldsymbol{x})^{n-1} \mathcal{S}_f(1)}, \tag{4}$$

where $\mathcal{S}_f$ is the surface area of $\mathbb{L}_f$ and ϕ is a density on $\mathbb{R}^+$. Thus, we need to compute the surface area of an arbitrary L_p-nested unit sphere to obtain the partition function of Equation (4).

Proposition 4 (Volume and Surface of the L_p-nested Sphere) *Let f be an L_p-nested function and let I be the set of all multi-indices denoting the inner nodes of the tree structure associated with f. The volume $\mathcal{V}_f(R)$ and the surface $\mathcal{S}_f(R)$ of the L_p-nested sphere with radius R are given by*

$$\mathcal{V}_f(R) = \frac{R^n 2^n}{n} \prod_{I \in \mathcal{I}} \left(\frac{1}{p_I^{\ell_I - 1}} \prod_{k=1}^{\ell_I - 1} B\left[\frac{\sum_{i=1}^{k} n_{I,k}}{p_I}, \frac{n_{I,k+1}}{p_I} \right] \right) \tag{5}$$

$$= \frac{R^n 2^n}{n} \prod_{I \in \mathcal{I}} \frac{\prod_{k=1}^{\ell_I} \Gamma\left[\frac{n_{I,k}}{p_I}\right]}{p_I^{\ell_I - 1} \Gamma\left[\frac{n_I}{p_I}\right]}, \tag{6}$$

$$\mathcal{S}_f(R) = R^{n-1} 2^n \prod_{I \in \mathcal{I}} \left(\frac{1}{p_I^{\ell_I - 1}} \prod_{k=1}^{\ell_I - 1} B\left[\frac{\sum_{i=1}^{k} n_{I,k}}{p_I}, \frac{n_{I,k+1}}{p_I} \right] \right) \tag{7}$$

$$= R^{n-1} 2^n \prod_{I \in \mathcal{I}} \frac{\prod_{k=1}^{\ell_I} \Gamma\left[\frac{n_{I,k}}{p_I}\right]}{p_I^{\ell_I - 1} \Gamma\left[\frac{n_I}{p_I}\right]} \tag{8}$$

where $B[a,b] = \frac{\Gamma[a]\Gamma[b]}{\Gamma[a+b]}$ denotes the β-function.

Proof The proof can be found in the Appendix B. ■

Inserting the surface area in Equation 4, we obtain the general form of an L_p-nested symmetric distribution for any given radial density ϕ.

Corollary 5 (L_p-nested Symmetric Distribution) *Let f be an L_p-nested function and ϕ a density on $\mathbb{R}^+$. The corresponding L_p-nested symmetric distribution is given by*

$$\rho(\boldsymbol{x}) = \frac{\phi(f(\boldsymbol{x}))}{f(\boldsymbol{x})^{n-1} \mathcal{S}_f(1)}$$

$$= \frac{\phi(f(\boldsymbol{x}))}{2^n f(\boldsymbol{x})^{n-1}} \prod_{I \in \mathcal{I}} \left(p_I^{\ell_I - 1} \prod_{k=1}^{\ell_I - 1} B\left[\frac{\sum_{i=1}^{k} n_{I,k}}{p_I}, \frac{n_{I,k+1}}{p_I} \right]^{-1} \right). \tag{9}$$

The results of Fernandez et al. (1995) imply that for any ν-spherically symmetric distribution, the radial part is independent of the directional part, that is, r is independent of $\boldsymbol{u}$. The distribution of $\boldsymbol{u}$ is entirely determined by the choice of ν, or by the L_p-nested function f in our case. The distribution of r is determined by the radial density ϕ. Together, an L_p-nested symmetric distribution is determined by both, the L_p-nested function f and the choice of ϕ. From Equation (9), we can see that its density function must be the inverse of the surface area of $\mathbb{L}_f$ times the radial density when transforming (4) into the coordinates of Definition 1 and separating r and $\boldsymbol{u}$ (the factor $f(\boldsymbol{x})^{n-1} = r$ cancels due to the determinant of the Jacobian). For that reason we call the distribution of $\boldsymbol{u}$ *uniform on the L_p-sphere* $\mathbb{L}_f$ in analogy to Song and Gupta (1997). Next, we state its form in terms of the coordinates $\boldsymbol{u}$.

Proposition 6 (L_p-nested Uniform Distribution) *Let f be an L_p-nested function. Let $\mathcal{L}$ be the set of multi-indices on the path from the root node to the leaf corresponding to x_n. The uniform*

distribution on the L_p-nested unit sphere, that is, the set $\mathbb{L}_f = \{\boldsymbol{x} \in \mathbb{R}^n | f(\boldsymbol{x}) = 1\}$ is given by the following density over $u_1, ..., u_{n-1}$

$$\rho(u_1, , ..., u_{n-1}) = \frac{\prod_{L \in \mathcal{L}} G_L(\boldsymbol{u}_{\widehat{L}})}{2^{n-1}} \prod_{I \in \mathcal{I}} \left(p_I^{\ell_I - 1} \prod_{k=1}^{\ell_I - 1} B\left[\frac{\sum_{i=1}^{k} n_{I,k}}{p_I}, \frac{n_{I,k+1}}{p_I} \right]^{-1} \right).$$

Proof Since the L_p-nested sphere is a measurable and compact set, the density of the uniform distribution is simply one over the surface area of the L_p-nested unit sphere. The surface $\mathcal{S}_f(1)$ is given by Proposition 4. Transforming $\frac{1}{\mathcal{S}_f(1)}$ into the coordinates of Definition 1 introduces the determinant of the Jacobian from Proposition 3 and an additional factor of 2 since the $(u_1, ..., u_{n-1}) \in \mathbb{R}^{n-1}$ have to account for both half-shells of the L_p-nested unit sphere, that is, to account for the fact that u_n could have been be positive or negative. This yields the expression above. ■

Example 4 *Let us again demonstrate the proposition at the special case where f is an L_p-norm $f(\boldsymbol{x}) = ||\boldsymbol{x}||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$. Using Proposition 4, the surface area is given by*

$$\mathcal{S}_{||\cdot||_p} = 2^n \frac{1}{p_\emptyset^{\ell_\emptyset - 1}} \prod_{k=1}^{\ell_\emptyset - 1} B\left[\frac{\sum_{i=1}^{k} n_k}{p_\emptyset}, \frac{n_{k+1}}{p_\emptyset} \right] = \frac{2^n \Gamma^n \left[\frac{1}{p}\right]}{p^{n-1} \Gamma\left[\frac{n}{p}\right]}.$$

The factor $G_n(\boldsymbol{u}_{\widehat{n}})$ is given by $\left(1 - \sum_{i=1}^{n-1} |u_i|^p\right)^{\frac{1-p}{p}}$ (see the L_p-norm example before), which, after including the factor 2, yields the uniform distribution on the L_p-sphere as defined in Song and Gupta (1997)

$$p(\boldsymbol{u}) = \frac{p^{n-1} \Gamma\left[\frac{n}{p}\right]}{2^{n-1} \Gamma^n \left[\frac{1}{p}\right]} \left(1 - \sum_{i=1}^{n-1} |u_i|^p\right)^{\frac{1-p}{p}}.$$

Example 5 *As a second illustrative example, we consider the uniform density on the L_p-nested unit ball, that is, the set $\{\boldsymbol{x} \in \mathbb{R}^n | f(\boldsymbol{x}) \leq 1\}$, and derive its radial distribution ϕ. The density of the uniform distribution on the unit L_p-nested ball does not depend on $\boldsymbol{x}$ and is given by $\rho(\boldsymbol{x}) = 1/\mathcal{V}_f(1)$. Transforming the density into the polar-like coordinates with the determinant from Proposition 3 yields*

$$\frac{1}{\mathcal{V}_f(1)} = \frac{n r^{n-1} \prod_{L \in \mathcal{L}} G_L(\boldsymbol{u}_{\widehat{L}})}{2^{n-1}} \prod_{I \in \mathcal{I}} \left(p_I^{\ell_I - 1} \prod_{k=1}^{\ell_I - 1} B\left[\frac{\sum_{i=1}^{k} n_{I,k}}{p_I}, \frac{n_{I,k+1}}{p_I} \right]^{-1} \right).$$

After separating out the uniform distribution on the L_p-nested unit sphere, we obtain the radial distribution

$$\phi(r) = n r^{n-1} \text{ for } 0 < r \leq 1$$

which is a β-distribution with parameters n and 1.

The radial distribution from the preceeding example is of great importance for our sampling scheme derived in Section 6. The idea behind it is the following: First, a sample from a "simple" L_p-nested symmetric distribution is drawn. Since the radial and the uniform component on the L_p-nested unit sphere are statistically independent, we can get a sample from the uniform distribution on the L_p-nested unit sphere by simply normalizing the sample from the simple distribution. Afterwards we can multiply it with a radius drawn from the radial distribution of the L_p-nested symmetric distribution that we actually want to sample from. The role of the simple distribution will be played by the uniform distribution within the L_p-nested unit ball. Sampling from it is basically done by applying the steps in Proposition 4's proof backwards. We lay out the sampling scheme in more detail in Section 6.

4. Marginals

In this section we discuss two types of marginals: First, we demonstrate that, in contrast to L_p-spherically symmetric distributions, marginals of L_p-nested symmetric distributions are not necessarily L_p-nested symmetric again. The second type of marginals we discuss are obtained by collapsing all leaves of a subtree into the value of the subtree's root node. For that case we derive an analytical expression and show that the values of the root node's children follow a special kind of Dirichlet distribution.

Gupta and Song (1997) show that marginals of L_p-spherically symmetric distributions are again L_p-spherically symmetric. This does not hold, however, for L_p-nested symmetric distributions. This can be shown by a simple counterexample. Consider the L_p-nested function

$$f(\boldsymbol{x}) = \left((|x_1|^{p_1} + |x_2|^{p_1})^{\frac{p_\emptyset}{p_1}} + |x_3|^{p_\emptyset} \right)^{\frac{1}{p_\emptyset}}.$$

The uniform distribution inside the L_p-nested ball corresponding to f is given by

$$\rho(\boldsymbol{x}) = \frac{n p_1 p_\emptyset \Gamma\left[\frac{2}{p_1}\right] \Gamma\left[\frac{3}{p_\emptyset}\right]}{2^3 \Gamma^2\left[\frac{1}{p_1}\right] \Gamma\left[\frac{2}{p_0}\right] \Gamma\left[\frac{1}{p_0}\right]}.$$

The marginal $\rho(x_1, x_3)$ is given by

$$\rho(x_1, x_3) = \frac{n p_1 p_\emptyset \Gamma\left[\frac{2}{p_1}\right] \Gamma\left[\frac{3}{p_\emptyset}\right]}{2^3 \Gamma^2\left[\frac{1}{p_1}\right] \Gamma\left[\frac{2}{p_0}\right] \Gamma\left[\frac{1}{p_0}\right]} \left((1 - |x_3|^{p_\emptyset})^{\frac{p_1}{p_\emptyset}} - |x_1|^{p_1} \right)^{\frac{1}{p_1}}.$$

This marginal is not L_p-spherically symmetric. Since any L_p-nested symmetric distribution in two dimensions must be L_p-spherically symmetric, it cannot be L_p-nested symmetric as well. Figure 3 shows a scatter plot of the marginal distribution. Besides the fact that the marginals are not contained in the family of L_p-nested symmetric distributions, it is also hard to derive a general form for them. This is not surprising given that the general form of marginals for L_p-spherically symmetric distributions involves an integral that cannot be solved analytically in general and is therefore not very useful in practice (Gupta and Song, 1997). For that reason we cannot expect marginals of L_p-nested symmetric distributions to have a simple form.

In contrast to single marginals, it is possible to specify the joint distribution of leaves and inner nodes of an L_p-nested tree if all descendants of their inner nodes in question have been integrated

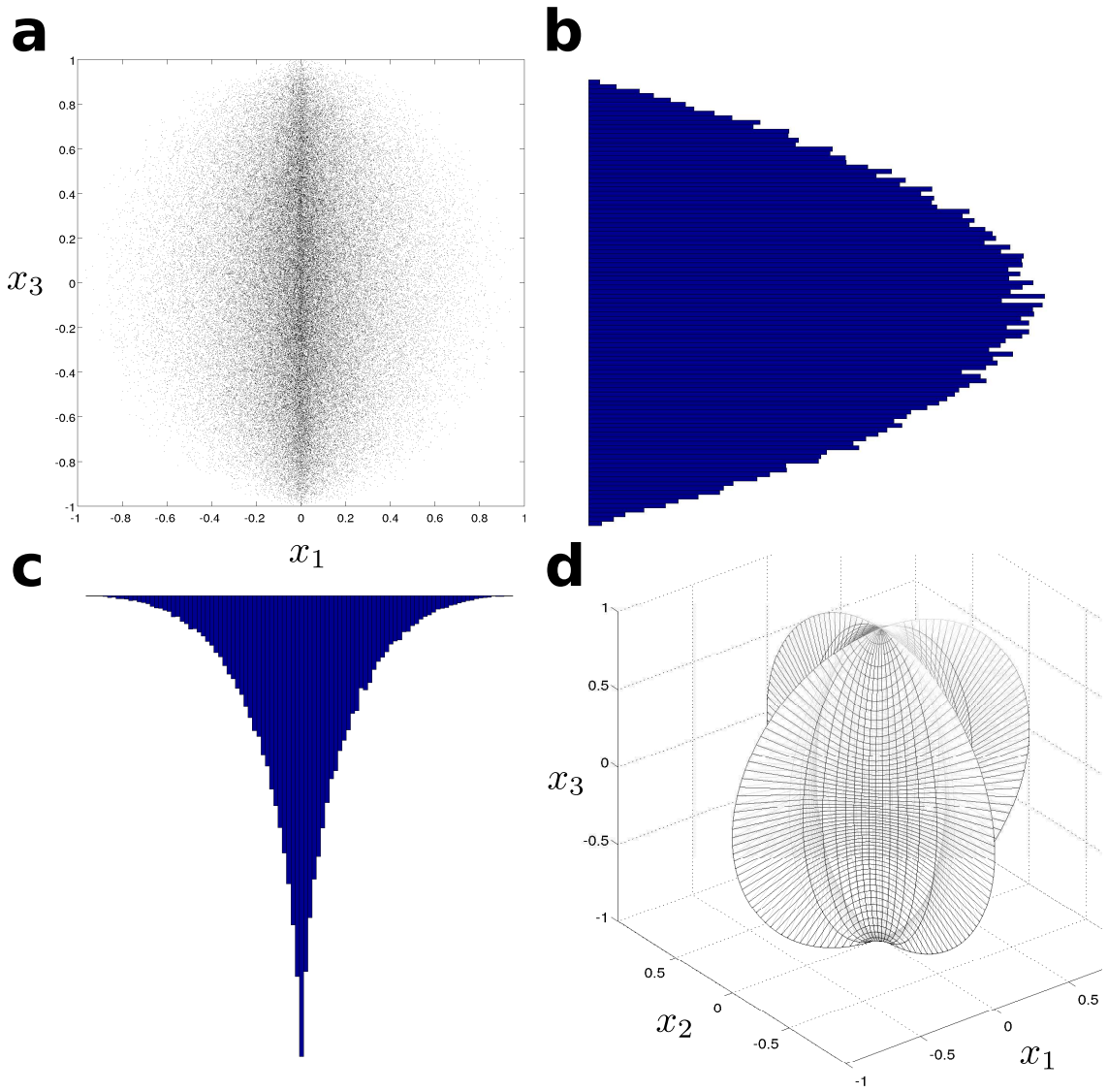


Figure 3: Marginals of L_p-nested symmetric distributions are not necessarily L_p-nested symmetric: Figure (**a**) shows a scatter plot of the (x_1, x_2)-marginal of the counterexample in the text with $p_\emptyset = 2$ and $p_1 = \frac{1}{2}$. Figure (**d**) displays the corresponding L_p-nested sphere. (**b-c**) show the univariate marginals for the scatter plot. Since any two-dimensional L_p-nested symmetric distribution must be L_p-spherically symmetric, the marginals should be identical. This is clearly not the case. Thus, (**a**) is not L_p-nested symmetric.

out. For the simple function above (the same that has been used in Example 1), the joint distribution of x_3 and $v_1 = \|(x_1, x_2)^\top\|_{p_1}$ would be an example of such a marginal. Since marginalization affects

the L_p-nested tree vertically, we call this type of marginals *layer marginals*. In the following, we present their general form.

From the form of a general L_p-nested function and the corresponding symmetric distribution, one might think that the layer marginals are L_p-nested symmetric again. However, this is not the case since the distribution over the L_p-nested unit sphere would deviate from the uniform distribution in most cases if the distribution of its children were L_p-spherically symmetric.

Proposition 7 *Let f be an L_p-nested function. Suppose we integrate out complete subtrees from the tree associated with f, that is, we transform subtrees into radial times uniform variables and integrate out the latter. Let $\mathcal{J}$ be the set of multi-indices of those nodes that have become new leaves, that is, whose subtrees have been removed, and let n_J be the number of leaves (in the original tree) in the subtree under the node J. Let $\boldsymbol{x}_{\widehat{\mathcal{J}}} \in \mathbb{R}^m$ denote those coefficients of $\boldsymbol{x}$ that are still part of that smaller tree and let $\boldsymbol{v}_{\mathcal{J}}$ denote the vector of inner nodes that became new leaves. The joint distribution of $\boldsymbol{x}_{\widehat{\mathcal{J}}}$ and $\boldsymbol{v}_{\mathcal{J}}$ is given by*

$$\rho(\boldsymbol{x}_{\widehat{\mathcal{J}}}, \boldsymbol{v}_{\mathcal{J}}) = \frac{\phi(f(\boldsymbol{x}_{\widehat{\mathcal{J}}}, \boldsymbol{v}_{\mathcal{J}}))}{S_f(f(\boldsymbol{x}_{\widehat{\mathcal{J}}}, \boldsymbol{v}_{\mathcal{J}}))} \prod_{J \in \mathcal{J}} v_J^{n_J - 1}. \tag{10}$$

Proof The proof can be found in the Appendix C. ■

Equation (10) has an interesting special case when considering the joint distribution of the root node's children.

Corollary 8 *The children of the root node $\boldsymbol{v}_{1:\ell_\emptyset} = (v_1, ..., v_{\ell_\emptyset})^\top$ follow the distribution*

$$\rho(\boldsymbol{v}_{1:\ell_\emptyset}) = \frac{p_\emptyset^{\ell_\emptyset - 1} \Gamma\left[\frac{n}{p_\emptyset}\right]}{f(v_1, ..., v_{\ell_\emptyset})^{n-1} 2^m \prod_{k=1}^{\ell_\emptyset} \Gamma\left[\frac{n_k}{p_\emptyset}\right]} \phi\left(f(v_1, ..., v_{\ell_\emptyset})\right) \prod_{i=1}^{\ell_\emptyset} v_i^{n_i - 1}$$

where $m \leq \ell_\emptyset$ is the number of leaves directly attached to the root node. In particular, $\boldsymbol{v}_{1:\ell_\emptyset}$ can be written as the product RU, where R is the L_p-nested radius and the single $|U_i|^{p_\emptyset}$ are Dirichlet distributed, that is, $(|U_1|^{p_\emptyset}, ..., |U_{\ell_\emptyset}|^{p_\emptyset}) \sim Dir\left[\frac{n_1}{p_\emptyset}, ..., \frac{n_{\ell_\emptyset}}{p_\emptyset}\right]$.

Proof The joint distribution is simply the application of Proposition (7). Note that $f(v_1, ..., v_{\ell_\emptyset}) = ||\boldsymbol{v}_{1:\ell_\emptyset}||_{p_\emptyset}$. Applying the pointwise transformation $s_i = |u_i|^{p_\emptyset}$ yields

$$(|U_1|^{p_\emptyset}, ..., |U_{\ell_\emptyset - 1}|^{p_\emptyset}) \sim \text{Dir}\left[\frac{n_1}{p_\emptyset}, ..., \frac{n_{\ell_\emptyset}}{p_\emptyset}\right].$$

■

The Corollary shows that the values $f_I(\boldsymbol{x}_I)$ at inner nodes I, in particular the ones directly below the root node, deviate considerably from L_p-spherical symmetry. If they were L_p-spherically symmetric, the $|U_i|^p$ should follow a Dirichlet distribution with parameters $\alpha_i = \frac{1}{p}$ as has been already shown by Song and Gupta (1997). The Corollary is a generalization of their result.

We can use the Corollary to prove an interesting fact about L_p-nested symmetric distributions: The only factorial L_p-nested symmetric distribution must be L_p-spherically symmetric.

Proposition 9 *Let $\boldsymbol{x}$ be L_p-nested symmetric distributed with independent marginals. Then $\boldsymbol{x}$ is L_p-spherically symmetric distributed. In particular, $\boldsymbol{x}$ follows a p-generalized Normal distribution.*

Proof The proof can be found in the Appendix D. ■

One immediate implication of Proposition 9 is that there is no factorial probability model corresponding to mixed norm regularizers which have the form $\sum_{i=1}^{k} \|\boldsymbol{x}_{I_k}\|_p^q$ where the index sets I_k form a partition of the dimensions $1,...,n$ (see, e.g., Zhao et al., 2008; Yuan and Lin, 2006; Kowalski et al., 2008). Many machine learning algorithms are equivalent to minimizing the sum of a regularizer $R(\boldsymbol{w})$ and a loss function $L(\boldsymbol{w},\boldsymbol{x}_1,...,\boldsymbol{x}_m)$ over the coefficient vector $\boldsymbol{w}$. If the $\exp(-R(\boldsymbol{w}))$ and $\exp(-L(\boldsymbol{w},\boldsymbol{x}_1,...,\boldsymbol{x}_m))$ correspond to normalizeable density models, the minimizing solution of the objective function can be seen as the maximum a posteriori (MAP) estimate of the posterior $p(\boldsymbol{w}|\boldsymbol{x}_1,...,\boldsymbol{x}_m) \propto p(\boldsymbol{w}) \cdot p(\boldsymbol{x}_1,...,\boldsymbol{x}_m|\boldsymbol{w}) = \exp(-R(\boldsymbol{w})) \cdot \exp(-L(\boldsymbol{w},\boldsymbol{x}_1,...,\boldsymbol{x}_m))$. In that sense, the regularizer naturally corresponds to the prior and the loss function corresponds to the likelihood. Very often, regularizers are specified as a norm over the coefficient vector $\boldsymbol{w}$ which in turn correspond to certain priors. For example, in Ridge regression (Hoerl, 1962) the coefficients are regularized via $\|\boldsymbol{w}\|_2^2$ which corresponds to a factorial zero mean Gaussian prior on $\boldsymbol{w}$. The L_1-norm $\|\boldsymbol{w}\|_1$ in the LASSO estimator (Tibshirani, 1996), again, is equivalent to a factorial Laplacian prior on $\boldsymbol{w}$. Like in these two examples, regularizers often correspond to a *factorial* prior.

Mixed norm regularizers naturally correspond to L_p-nested symmetric distributions. Proposition 9 shows that there is no factorial prior that corresponds to such a regularizer. In particular, it implies that the prior cannot be factorial between groups and coefficients at the same time. This means that those regularizers implicitly assume statistical dependencies between the coefficient variables. Interestingly, for $q = 1$ and $p = 2$ the intuition behind these regularizers is exactly that whole groups I_k get switched on at once, but the groups are sparse. The Proposition shows that this might not only be due to sparseness but also due to statistical dependencies between the coefficients within one group. The L_p-nested symmetric distribution which implements independence between groups will be further discussed below as a generalization of the p-generalized Normal (see Section 8). Note that the marginals can be independent if the regularizer is of the form $\sum_{i=1}^{k} \|\boldsymbol{x}_{I_k}\|_p^p$. However, in this case $p = q$ and the L_p-nested function collapses to a simple L_p-norm which means that the regularizer is not mixed norm.

5. Maximum Likelihood Estimation of L_p-Nested Symmetric Distributions

In this section, we describe procedures for maximum likelihood fitting of L_p-nested symmetric distributions on data. We provide a toolbox online for fitting L_p-spherically symmetric and L_p-nested symmetric distributions to data. The toolbox can be downloaded at `http://www.kyb.tuebingen.mpg.de/bethge/code/`.

Depending on which parameters are to be estimated, the complexity of fitting an L_p-nested symmetric distribution varies. We start with the simplest case and later continue with more complex ones. Throughout this subsection, we assume that the model has the form $p(\boldsymbol{x}) = \rho(W\boldsymbol{x}) \cdot |\det W| = \frac{\phi(W\boldsymbol{x})}{f(W\boldsymbol{x})^{n-1} S_f(1)} \cdot |\det W|$ where $W \in \mathbb{R}^{n \times n}$ is a complete whitening matrix. This means that given any whitening matrix W_0, the freedom in fitting W is to estimate an orthonormal matrix $Q \in SO(n)$ such that $W = QW_0$. This is analogous to the case of elliptically contoured distributions where the

distributions can be endowed with 2nd-order correlations via W. In the following, we ignore the determinant of W since data points can always be rescaled such that $\det W = 1$.

The simplest case is to fit the parameters of the radial distribution when the tree structure, the values of the p_I, and W are fixed. Due to the special form of L_p-nested symmetric distributions (4), it then suffices to carry out maximum likelihood estimation on the radial component only, which renders maximum likelihood estimation efficient and robust. This is because the only remaining parameters are the parameters $\boldsymbol{\vartheta}$ of the radial distribution and, therefore,

$$\begin{aligned}\operatorname{argmax}_{\boldsymbol{\vartheta}} \log \rho(W\boldsymbol{x}|\boldsymbol{\vartheta}) &= \operatorname{argmax}_{\boldsymbol{\vartheta}} \left(-\log \mathcal{S}_f(f(W\boldsymbol{x})) + \log \phi(f(W\boldsymbol{x})|\boldsymbol{\vartheta})\right) \\ &= \operatorname{argmax}_{\boldsymbol{\vartheta}} \log \phi(f(W\boldsymbol{x})|\boldsymbol{\vartheta}).\end{aligned}$$

In a slightly more complex case, when only the tree structure and W are fixed, the values of the p_I, $I \in \mathcal{I}$ and $\boldsymbol{\vartheta}$ can be jointly estimated via gradient ascent on the log-likelihood. The gradient for a single data point $\boldsymbol{x}$ with respect to the vector $\boldsymbol{p}$ that holds all p_I for all $I \in \mathcal{I}$ is given by

$$\nabla_{\boldsymbol{p}} \log \rho(W\boldsymbol{x}) = \frac{d}{dr} \log \phi(f(W\boldsymbol{x})) \cdot \nabla_{\boldsymbol{p}} f(W\boldsymbol{x}) - \frac{(n-1)}{f(W\boldsymbol{x})} \nabla_{\boldsymbol{p}} f(W\boldsymbol{x}) - \nabla_{\boldsymbol{p}} \log \mathcal{S}_f(1).$$

For i.i.d. data points $\boldsymbol{x}_i$ the joint gradient is given by the sum over the gradients for the single data points. Each of them involves the gradient of f as well as the gradient of the log-surface area of $\mathbb{L}_f$ with respect to $\boldsymbol{p}$, which can be computed via the recursive equations

$$\frac{\partial}{\partial p_J} v_I = \begin{cases} 0 & \text{if } I \text{ is not a prefix of } J \\ v_I^{1-p_I} v_{I,k}^{p_I - 1} \cdot \frac{\partial}{\partial p_J} v_{I,k} & \text{if } I \text{ is a prefix of } J \\ \frac{v_J}{p_J} \left(v_J^{-p_J} \sum_{k=1}^{\ell_J} v_{J,k}^{p_J} \cdot \log v_{J,k} - \log v_J \right) & \text{if } J = I \end{cases}$$

and

$$\begin{aligned}\frac{\partial}{\partial p_J} \log \mathcal{S}_f(1) = &- \frac{\ell_J - 1}{p_J} + \sum_{k=1}^{\ell_J - 1} \Psi\left[\frac{\sum_{i=1}^{k+1} n_{J,k}}{p_J}\right] \frac{\sum_{i=1}^{k+1} n_{J,k}}{p_J^2} \\ &- \sum_{k=1}^{\ell_J - 1} \Psi\left[\frac{\sum_{i=1}^{k} n_{J,k}}{p_J}\right] \frac{\sum_{i=1}^{k} n_{J,k}}{p_J^2} - \sum_{k=1}^{\ell_J - 1} \Psi\left[\frac{n_{J,k+1}}{p_J}\right] \frac{n_{J,k+1}}{p_J^2},\end{aligned}$$

where $\Psi[t] = \frac{d}{dt} \log \Gamma[t]$ denotes the digamma function. When performing the gradient ascent, one needs to set $\mathbf{0}$ as a lower bound for $\boldsymbol{p}$. Note that, in general, this optimization might be a highly non-convex problem.

On the next level of complexity, only the tree structure is fixed, and W can be estimated along with the other parameters by joint optimization of the log-likelihood with respect to $\boldsymbol{p}$, $\boldsymbol{\vartheta}$ and W. Certainly, this optimization problem is also not convex in general. Usually, it is numerically more robust to whiten the data first with some whitening matrix W_0 and perform a gradient ascent on the special orthogonal group $SO(n)$ with respect to Q for optimizing $W = QW_0$. Given the gradient $\nabla_W \log \rho(W\boldsymbol{x})$ of the log-likelihood, the optimization can be carried out by performing line searches along geodesics as proposed by Edelman et al. (1999) (see also Absil et al. (2007)) or by projecting $\nabla_W \log \rho(W\boldsymbol{x})$ on the tangent space $T_W SO(n))$ and performing a line search along $SO(n)$ in that direction as proposed by Manton (2002).

The general form of the gradient to be used in such an optimization scheme can be defined as

$$\begin{aligned}&\nabla_W \log \rho(W\boldsymbol{x})\\ =&\nabla_W \left(-(n-1)\cdot \log f(W\boldsymbol{x}) + \log \phi(f(W\boldsymbol{x}))\right)\\ =&-\frac{(n-1)}{f(W\boldsymbol{x})}\cdot \nabla_{\boldsymbol{y}} f(W\boldsymbol{x})\cdot \boldsymbol{x}^\top + \frac{d\log\phi(r)}{dr}(f(W\boldsymbol{x}))\cdot \nabla_{\boldsymbol{y}} f(W\boldsymbol{x})\cdot \boldsymbol{x}^\top,\end{aligned}$$

where the derivatives of f with respect to $\boldsymbol{y}$ are defined by recursive equations

$$\frac{\partial}{\partial y_i} v_I = \begin{cases} 0 & \text{if } i \notin I \\ \operatorname{sgn} y_i & \text{if } v_{I,k} = |y_i| \\ v_I^{1-p_I}\cdot v_{I,k}^{p_I-1}\cdot \frac{\partial}{\partial y_i} v_{I,k} & \text{for } i \in I, k. \end{cases}$$

Note, that f might not be differentiable at $\boldsymbol{y} = 0$. However, we can always define a sub-derivative at zero, which is zero for $p_I \neq 1$ and $[-1,1]$ for $p_I = 1$. Again, the gradient for i.i.d. data points $\boldsymbol{x}_i$ is given by the sum over the single gradients.

Finally, the question arises whether it is possible to estimate the tree structure from data as well. A simple heuristic would be to start with a very large tree, for example, a full binary tree, and to prune out inner nodes for which the parents and the children have sufficiently similar values for their p_I. The intuition behind this is that if they were exactly equal, they would cancel in the L_p-nested function. This heuristic is certainly sub-optimal. Firstly, the optimization will be time consuming since there can be about as many p_I as there are leaves in the L_p-nested tree (a full binary tree on n dimensions will have $n-1$ inner nodes) and due to the repeated optimization after the pruning steps. Secondly, the heuristic does not cover all possible trees on n leaves. For example, if two leaves are separated by the root node in the original full binary tree, there is no way to prune out inner nodes such that the path between those two nodes will not contain the root node anymore.

The computational complexity for the estimation of all other parameters despite the tree structure is difficult to assess in general because they depend, for example, on the particular radial distribution used. While the maximum likelihood estimation of a simple log-Normal distribution only involves the computation of a mean and a variance which are in $O(m)$ for m data points, a mixture of log-Normal distributions already requires an EM algorithm which is computationally more expensive. Additionally, the time it takes to optimize the likelihood depends on the starting point as well as the convergence rate, and we neither have results about the convergence rate nor is it possible to make problem independent statements about a good initialization of the parameters. For this reason we state only the computational complexity of single steps involved in the optimization.

Computation of the gradient $\nabla_{\boldsymbol{p}} \log \rho(W\boldsymbol{x})$ involves the derivative of the radial distribution, the computation of the gradients $\nabla_{\boldsymbol{p}} f(W\boldsymbol{x})$ and $\nabla_{\boldsymbol{p}} \mathcal{S}_f(1)$. Assuming that the derivative of the radial distribution can be computed in $O(1)$ for each single data point, the costly steps are the other two gradients. Computing $\nabla_{\boldsymbol{p}} f(W\boldsymbol{x})$ basically involves visiting each node of the tree once and performing a constant number of operations for the local derivatives. Since every inner node in an L_p-nested tree must have at least two children, the worst case would be a full binary tree which has $2n-1$ nodes and leaves. Therefore, the gradient can be computed in $O(nm)$ for m data points. For similar reasons, $f(W\boldsymbol{x})$, $\nabla_{\boldsymbol{p}} \log \mathcal{S}_f(1)$, and the evaluation of the likelihood can also be computed in $O(nm)$. This means that each step in the optimization of $\boldsymbol{p}$ can be done $O(nm)$ plus the computational costs for the line search in the gradient ascent. When optimizing for $W = QW_0$ as well, the computational

costs per step increase to $O(n^3 + n^2 m)$ since m data points have to be multiplied with W at each iteration (requiring $O(n^2 m)$ steps), and the line search involves projecting Q back onto $SO(n)$ which requires an inverse matrix square root or a similar computation in $O(n^3)$.

For comparison, each step of fast ICA (Hyvärinen and O., 1997) for a complete demixing matrix takes $O(n^2 m)$ when using hierarchical orthogonalization and $O(n^2 m + n^3)$ for symmetric orthogonalization. The same applies to fitting an ISA model (Hyvärinen and Hoyer, 2000; Hyvärinen and Köster, 2006, 2007). A Gaussian Scale Mixture (GSM) model does not need to estimate another orthogonal rotation Q because it belongs to the class of spherically symmetric distributions and is, therefore, invariant under transformations from $SO(n)$ (Wainwright and Simoncelli, 2000). Therefore, fitting a GSM corresponds to estimating the parameters of the scale distribution which is $O(nm)$ in the best case but might be costlier depending on the choice of the scale distribution.

6. Sampling from L_p-Nested Symmetric Distributions

In this section, we derive a sampling scheme for arbitrary L_p-nested symmetric distributions which can for example be used for solving integrals when using L_p-nested symmetric distributions for Bayesian learning. Exact sampling from an arbitrary L_p-nested symmetric distribution is in fact straightforward due to the following observation: Since the radial and the uniform component are independent, normalizing a sample from any L_p-nested symmetric distribution to f-length one yields samples from the uniform distribution on the L_p-nested unit sphere. By multiplying those uniform samples with new samples from another radial distribution, one obtains samples from another L_p-nested symmetric distribution. Therefore, for each L_p-nested function f, a single L_p-nested symmetric distribution which can be easily sampled from is enough. Sampling from all other L_p-nested symmetric distributions with respect to f is then straightforward due to the method we just described. Gupta and Song (1997) sample from the p-generalized Normal distribution since it has independent marginals which makes sampling straightforward. Due to Proposition 9, no such factorial L_p-nested symmetric distribution exists. Therefore, a sampling scheme like that for L_p-spherically symmetric distributions is not applicable. Instead we choose to sample from the uniform distribution inside the L_p-nested unit ball for which we already computed the radial distribution in Example 5. The distribution has the form $\rho(\boldsymbol{x}) = \frac{1}{\mathcal{V}_f(1)}$. In order to sample from that distribution, we will first only consider the uniform distribution in the positive quadrant of the unit L_p-nested ball which has the form $\rho(\boldsymbol{x}) = \frac{2^n}{\mathcal{V}_f(1)}$. Samples from the uniform distributions inside the whole ball can be obtained by multiplying each coordinate of a sample with independent samples from the uniform distribution over $\{-1, 1\}$.

The idea of the sampling scheme for the uniform distribution inside the L_p-nested unit ball is based on the computation of the volume of the L_p-nested unit ball in Proposition 4. The basic mechanism underlying the sampling scheme below is to apply the steps of the proof backwards, which is based on the following idea: The volume of the L_p-unit ball can be computed by computing its volume on the positive quadrant only and multiplying the result with 2^n afterwards. The key is now to not transform the whole integral into radial and uniform coordinates at once, but successively upwards in the tree. We will demonstrate this through a brief example which also should make the sampling scheme below more intuitive. Consider the L_p-nested function

$$f(\boldsymbol{x}) = \left(|x_1|^{p_\emptyset} + \left(|x_2|^{p_1} + |x_3|^{p_1} \right)^{\frac{p_\emptyset}{p_1}} \right)^{\frac{1}{p_\emptyset}}.$$

To solve the integral

$$\int_{\{\boldsymbol{x}: f(\boldsymbol{x}) \leq 1 \,\&\, \boldsymbol{x} \in \mathbb{R}_+^n\}} d\boldsymbol{x},$$

we first transform x_2 and x_3 into radial and uniform coordinates only. According to Proposition 3 the determinant of the mapping $(x_2, x_3) \mapsto (v_1, \tilde{u}) = (\|\boldsymbol{x}_{2:3}\|_{p_1}, \boldsymbol{x}_{2:3} / \|\boldsymbol{x}_{2:3}\|_{p_1})$ is given by $v_1 (1 - \tilde{u}^{p_1})^{\frac{1-p_1}{p_1}}$. Therefore the integral transforms into

$$\int_{\{\boldsymbol{x}: f(\boldsymbol{x}) \leq 1 \,\&\, \boldsymbol{x} \in \mathbb{R}_+^n\}} d\boldsymbol{x} = \int_{\{v_1, x_1 : f(x_1, v_1) \leq 1 \,\&\, x_1, v_1 \in \mathbb{R}_+\}} \int \int v_1 (1 - \tilde{u}^{p_1})^{\frac{1-p_1}{p_1}} dx_1 dv_1 d\tilde{u}.$$

Now we can separate the integrals over x_1 and v_1, and the integral over $\tilde{u}$, since the boundary of the outer integral does only depend on v_1 and not on $\tilde{u}$:

$$\int_{\{\boldsymbol{x}: f(\boldsymbol{x}) \leq 1 \,\&\, \boldsymbol{x} \in \mathbb{R}_+^n\}} d\boldsymbol{x} = \int (1 - \tilde{u}^{p_1})^{\frac{1-p_1}{p_1}} d\tilde{u} \cdot \int_{\{v_1, x_1 : f(x_1, v_1) \leq 1 \,\&\, x_1, v_1 \in \mathbb{R}_+\}} \int v_1 dx_1 dv_1.$$

The value of the first integral is known explicitly since the integrand equals the uniform distribution on the $\|\cdot\|_{p_1}$-unit sphere. Therefore, the value of the integral must be its normalization constant which we can get using Proposition 4:

$$\int (1 - \tilde{u}^{p_1})^{\frac{1-p_1}{p_1}} d\tilde{u} = \frac{\Gamma\left[\frac{1}{p_1}\right]^2 \cdot p_1}{\Gamma\left[\frac{2}{p_1}\right]}.$$

An alternative way to arrive at this result is to use the transformation $s = \tilde{u}^{p_1}$ and to notice that the integrand is a Dirichlet distribution with parameters $\alpha_i = \frac{1}{p_1}$. The normalization constant of the Dirichlet distribution and the constants from the determinant of the Jacobian of the transformation yield the same result.

To compute the remaining integral, the same method can be applied again yielding the volume of the L_p-nested unit ball. The important part for the sampling scheme, however, is not the volume itself but the fact that the intermediate results in this integration process equal certain distributions. As shown in Example 5 the radial distribution of the uniform distribution on the unit ball is $\beta[n, 1]$, and as just indicated by the example above, the intermediate results can be seen as transformed variables from a Dirichlet distribution. This fact holds true even for more complex L_p-nested unit balls although the parameters of the Dirichlet distribution can be slightly different. Reversing the steps leads us to the following sampling scheme. First, we sample from the β-distribution which gives us the radius $v_\emptyset$ on the root node. Then we sample from the appropriate Dirichlet distribution and exponentiate the samples by $\frac{1}{p_\emptyset}$ which transforms them into the analogs of the variable u from above. Scaling the result with the sample $v_\emptyset$ yields the values of the root node's children, that is, the analogs of x_1 and v_1. Those are the new radii for the levels below them where we simply repeat this procedure with the appropriate Dirichlet distributions and exponents. The single steps are summarized in Algorithm 1.

The computational complexity of the sampling scheme is $O(n)$. Since the sampling procedure is like expanding the tree node by node starting with the root, the number of inner nodes and leaves is the total number of samples that have to be drawn from Dirichlet distributions. Every node in an L_p-nested tree must at least have two children. Therefore, the maximal number of inner nodes and leaves is $2n - 1$ for a full binary tree. Since sampling from a Dirichlet distribution is also in $O(n)$, the total computational complexity for one sample is in $O(n)$.

Algorithm 1 Exact sampling algorithm for L_p-nested symmetric distributions

Input: The radial distribution ϕ of an L_p-nested symmetric distribution ρ for the L_p-nested function f.
Output: Sample $\boldsymbol{x}$ from ρ.
Algorithm

1. Sample $v_\emptyset$ from a beta distribution $\beta[n,1]$.
2. For each inner node I of the tree associated with f, sample the auxiliary variable $\boldsymbol{s}_I$ from a Dirichlet distribution $\mathrm{Dir}\left[\frac{n_{I,1}}{p_I},...,\frac{n_{I,\ell_I}}{p_I}\right]$ where $n_{I,k}$ are the number of leaves in the subtree under node I,k. Obtain coordinates on the L_p-nested sphere within the positive orthant by $\boldsymbol{s}_I \mapsto \boldsymbol{s}_I^{\frac{1}{p_I}} = \tilde{\boldsymbol{u}}_I$ (the exponentiation is taken component-wise).
3. Transform these samples to Cartesian coordinates by $v_I \cdot \tilde{\boldsymbol{u}}_I = \boldsymbol{v}_{I,1:\ell_I}$ for each inner node, starting from the root node and descending to lower layers. The components of $\boldsymbol{v}_{I,1:\ell_I}$ constitute the radii for the layer direct below them. If $I = \emptyset$, the radius had been sampled in step 1.
4. Once the two previous steps have been repeated until no inner node is left, we have a sample $\boldsymbol{x}$ from the uniform distribution in the positive quadrant. Normalize $\boldsymbol{x}$ to get a uniform sample from the sphere $\boldsymbol{u} = \frac{\boldsymbol{x}}{f(\boldsymbol{x})}$.
5. Sample a new radius $\tilde{v}_\emptyset$ from the radial distribution of the target radial distribution ϕ and obtain the sample via $\tilde{\boldsymbol{x}} = \tilde{v}_\emptyset \cdot \boldsymbol{u}$.
6. Multiply each entry x_i of $\tilde{\boldsymbol{x}}$ by an independent sample z_i from the uniform distribution over $\{-1,1\}$.

7. Robust Bayesian Inference of the Location

For L_p-spherically symmetric distributions with a location and a scale parameter

$$p(\boldsymbol{x}|\boldsymbol{\mu},\tau) = \tau^n \rho(\|\tau(\boldsymbol{x}-\boldsymbol{\mu})\|_p),$$

Osiewalski and Steel (1993) derived the posterior in closed form using a prior $p(\boldsymbol{\mu},\tau) = p(\mu)\cdot c\cdot\tau^{-1}$, and showed that $p(\boldsymbol{x},\boldsymbol{\mu})$ does not depend on the radial distribution ϕ, that is, the particular type of L_p-spherically symmetric distributions used for a fixed p. The prior on τ corresponds to an improper Jeffrey's prior which is used to represent lack of prior knowledge on the scale. The main implication of their result is that Bayesian inference of the location $\boldsymbol{\mu}$ under that prior on the scale does not depend on the particular type of L_p-spherically symmetric distribution used for inference. This means that under the assumption of an L_p-spherically symmetric distributed variable, for a fixed p, one has to know the exact form of the distribution in order to compute the location parameter.

It is straightforward to generalize their result to L_p-nested symmetric distributions and, hence, making it applicable to a larger class of distributions. Note that when using any L_p-nested symmetric distribution, introducing a scale and a location via the transformation $\boldsymbol{x} \mapsto \tau(\boldsymbol{x}-\boldsymbol{\mu})$ introduces a factor of τ^n in front of the distribution.

Proposition 10 *For fixed values $p_\emptyset, p_1, \ldots$ and two independent priors $p(\boldsymbol{\mu},\tau) = p(\boldsymbol{\mu}) \cdot c\tau^{-1}$ of the location μ and the scale τ where the prior on τ is an improper Jeffrey's prior, the joint distribution $p(\boldsymbol{x},\boldsymbol{\mu})$ is given by*

$$p(\boldsymbol{x},\boldsymbol{\mu}) = f(\boldsymbol{x}-\boldsymbol{\mu})^{-n} \cdot c \cdot \frac{1}{Z} \cdot p(\boldsymbol{\mu}),$$

where Z denotes the normalization constant of the L_p-nested uniform distribution.

Proof Given any L_p-nested symmetric distribution $\rho(f(\boldsymbol{x}))$, the transformation into the polar-like coordinates yields the following relation

$$1 = \int \rho(f(\boldsymbol{x}))d\boldsymbol{x} = \int\int \prod_{L\in\mathcal{L}} G_L(\boldsymbol{u}_{\widehat{L}}) r^{n-1}\rho(r)drd\boldsymbol{u} = \int \prod_{L\in\mathcal{L}} G_L(\boldsymbol{u}_{\widehat{L}})d\boldsymbol{u} \cdot \int r^{n-1}\rho(r)dr.$$

Since $\prod_{L\in\mathcal{L}} G_L(\boldsymbol{u}_{\widehat{L}})$ is the unnormalized uniform distribution on the L_p-nested unit sphere, the integral must equal the normalization constant which we denote with Z for brevity (see Proposition 6 for an explicit expression). This implies that ρ has to fulfill

$$\frac{1}{Z} = \int r^{n-1}\rho(r)dr.$$

Writing down the joint distribution of $\boldsymbol{x},\boldsymbol{\mu}$ and τ, and using the substitution $s = \tau f(\boldsymbol{x}-\boldsymbol{\mu})$ we obtain

$$\begin{aligned} p(\boldsymbol{x},\boldsymbol{\mu}) &= \int \tau^n \rho(f(\tau(\boldsymbol{x}-\boldsymbol{\mu}))) \cdot c\tau^{-1} \cdot p(\boldsymbol{\mu})d\tau \\ &= \int s^{n-1}\rho(s) \cdot c \cdot p(\boldsymbol{\mu}) f(\boldsymbol{x}-\boldsymbol{\mu})^{-n} ds \\ &= f(\boldsymbol{x}-\boldsymbol{\mu})^{-n} \cdot c \cdot \frac{1}{Z} \cdot p(\boldsymbol{\mu}). \end{aligned}$$

■

Note that this result could easily be extended to ν-spherical distributions. However, in this case the normalization constant Z cannot be computed for most cases and, therefore, the posterior would not be known explicitly.

8. Relations to ICA, ISA and Over-Complete Linear Models

In this section, we explain the relations among L_p-spherically symmetric, L_p-nested symmetric, ICA and ISA models. For a general overview see Figure 4.

The density model underlying ICA models the joint distribution of the signal $\boldsymbol{x}$ as a linear superposition of statistically independent hidden sources $A\boldsymbol{y} = \boldsymbol{x}$ or $\boldsymbol{y} = W\boldsymbol{x}$. If the marginals of the hidden sources belong to the exponential power family, we obtain the p-generalized Normal which is a subset of the L_p-spherically symmetric class. The p-generalized Normal distribution $p(\boldsymbol{y}) \propto \exp(-\tau\|\boldsymbol{y}\|_p^p)$ is a density model that is often used in ICA algorithms for kurtotic natural signals like images and sound by optimizing a demixing matrix W w.r.t. to the model $p(\boldsymbol{y}) \propto \exp(-\tau\|W\boldsymbol{x}\|_p^p)$ (Lee and Lewicki, 2000; Zhang et al., 2004; Lewicki, 2002). It can be

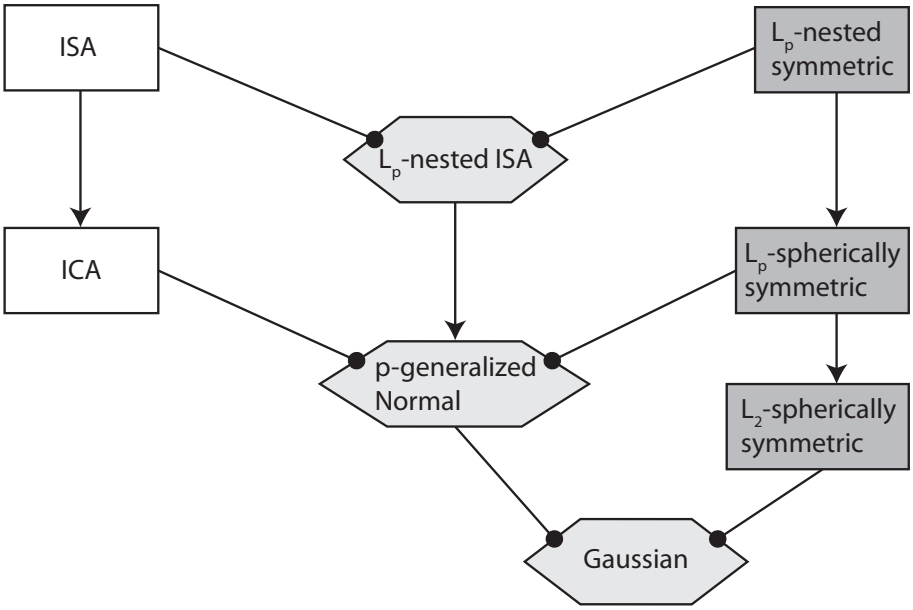


Figure 4: Relations between the different classes of distributions: Arrows indicate that the child class is a specialization (subset) of the parent class. Polygon-shaped classes are intersections of those parent classes which are connected via edges with round arrow-heads. For one-dimensional subspaces ISA is a superclass of ICA. All classes belonging to ISA are colored white or light gray. L_p-nested symmetric distributions are a superclass of L_p-spherically symmetric distributions. All L_p-nested symmetric models are colored dark or light gray. L_p-nested ISA models live in the intersection of L_p-nested symmetric distributions and ISA models. Those L_p-nested ISA models that are L_p-spherically symmetric are also ICA models: This is the class of p-generalized Normal distributions. If p is fixed to two, one obtains the L_2-spherically symmetric distributions. The only class of distributions in the intersection between spherically symmetric distributions and ICA models is the Gaussian.

shown that the p-generalized Normal is the only factorial model in the class of L_p-spherically symmetric models (Sinz et al., 2009a), and, by Proposition 9, also the only factorial L_p-nested symmetric distribution.

An important generalization of ICA is the independent subspace analysis (ISA) proposed by Hyvärinen and Hoyer (2000) and by Hyvärinen and Köster (2007) who used L_p-spherically symmetric distributions to model the single subspaces, that is, each ρ_k below was L_p-spherically symmetric. Like in ICA, ISA models the hidden sources of the signal as a product of multivariate distributions:

$$\rho(\boldsymbol{y}) = \prod_{k=1}^{K} \rho_k(\boldsymbol{y}_{I_k}).$$

Here, $\boldsymbol{y} = W\boldsymbol{x}$ and I_k are index sets selecting the different subspaces from the responses of W to $\boldsymbol{x}$. The collection of index sets I_k forms a partition of $1,\ldots,n$. ICA is a special case of ISA in which

$I_k = \{k\}$ such that all subspaces are one-dimensional. For the ISA models used by Hyvärinen et al. the distribution on the subspaces was chosen to be either spherically or L_p-spherically symmetric.

In its general form, ISA is not a generalization of L_p-spherically symmetric distributions. The most general ISA model for the transformed data $\boldsymbol{y} = W\boldsymbol{x}$ does not assume a certain type of distribution on the single subspace like in Hyvärinen and Köster (2007). While one could say for any non-factorial distribution that a factorial product over subspaces is a generalization, this is certainly a trivial step. Only in this particular sense is the particular ISA model by Hyvärinen and Köster (2007) a generalization of L_p-spherically symmetric distributions.

In contrast to ISA, L_p-nested symmetric distributions generally do not make an independence assumption on the "subspaces". In fact, for most of the models the subspaces will be dependent (see also our diagram in Figure 4). Therefore, not every ISA model is automatically L_p-nested symmetric and vice versa. In fact, in Sinz et al. (2009b) we have demonstrated for natural images that the dependencies *between* subspaces is stronger than the dependencies *within* subspaces on natural image patches. This is in stark contrast to the assumptions underlying ISA.

Note also that the product of L_p-spherically symmetric distributions used by Hyvärinen and Köster (2007) is not an L_p-nested function (Equation (6) in Hyvärinen and Köster, 2007) since the single a_j can be different and, therefore, the overall function is not positively homogeneous in general.

ICA and ISA have been used to infer features from natural signals, in particular from natural images. However, as mentioned by several authors (Zetzsche et al., 1993; Simoncelli, 1997; Wainwright and Simoncelli, 2000) and demonstrated quantitatively by Bethge (2006) and Eichhorn et al. (2009), the assumptions underlying linear ICA are not well matched by the statistics of the pixel intensities of natural images. A reliable parametric way to assess how well the independence assumption is met by a signal at hand is to fit a more general class of distributions that contains factorial as well as non-factorial distributions which both can equally well reproduce the marginals. By comparing the likelihood on held out test data between the best fitting non-factorial and the best-fitting factorial case, one can assess how well the sources can be described by a factorial distribution. For natural images, for example, one can use an arbitrary L_p-spherically symmetric distribution $\rho(\|W\boldsymbol{x}\|_p)$, fit it to the whitened data and compare its likelihood on held out test data to the one of the p-generalized Normal distribution (Sinz and Bethge, 2009). Since any choice of radial distribution ϕ determines a particular L_p-spherically symmetric distribution, the idea is to explore the space between factorial and non-factorial models by using a very flexible density ϕ on the radius. Note that having an explicit expression of the normalization constant allows for particularly reliable model comparisons via the likelihood. For many graphical models, for instance, such an explicit and computable expression is often not available.

The same type of dependency-analysis can be carried out for ISA using L_p-nested symmetric distributions (Sinz et al., 2009b). Figure 5 shows the L_p-nested tree corresponding to an ISA with four subspaces. In general, for such trees, each inner node—except the root node—corresponds to a single subspace. When using the radial distribution

$$\phi_\emptyset(v_\emptyset) = \frac{p_\emptyset v_\emptyset^{n-1}}{\Gamma\left[\frac{n}{p_\emptyset}\right] s^{\frac{n}{p_\emptyset}}} \exp\left(-\frac{v_\emptyset^{p_\emptyset}}{s}\right), \tag{11}$$

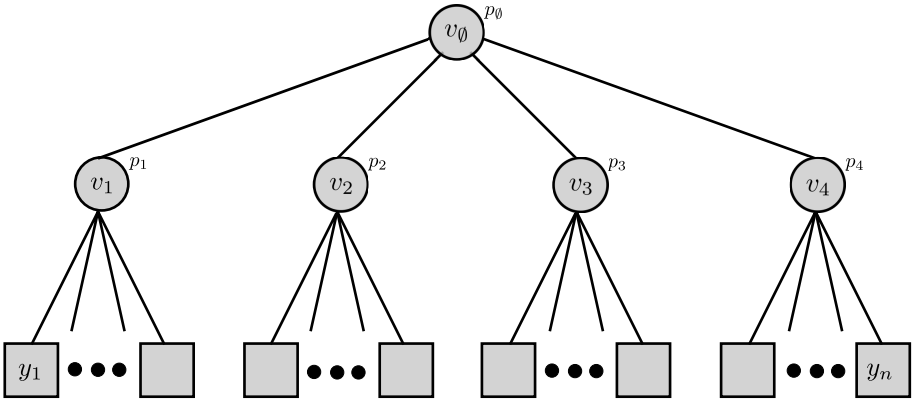


Figure 5: Tree corresponding to an L_p-nested ISA model.

the subspaces $v_1, ..., v_{\ell_\emptyset}$ become independent and one obtains an ISA model of the form

$$\begin{aligned}
\rho(\boldsymbol{y}) &= \frac{1}{Z}\exp\left(-\frac{f(\boldsymbol{y})^{p_\emptyset}}{s}\right) \\
&= \frac{1}{Z}\exp\left(-\frac{\sum_{k=1}^{\ell_\emptyset}\|\boldsymbol{y}_{I_k}\|_{p_k}}{s}\right) \\
&= \frac{p_\emptyset^{\ell_\emptyset}}{s^{\frac{n}{p_\emptyset}}\prod_{i=1}^{\ell_\emptyset}\Gamma\left[\frac{n_i}{p_\emptyset}\right]}\exp\left(-\frac{\sum_{k=1}^{\ell_\emptyset}\|\boldsymbol{y}_{I_k}\|_{p_k}}{s}\right)\prod_{k=1}^{\ell_\emptyset}\frac{p_k^{\ell_k-1}\Gamma\left[\frac{n_k}{p_k}\right]}{2^{n_k}\Gamma^{n_k}\left[\frac{1}{p_I}\right]},
\end{aligned}$$

which has L_p-spherically symmetric distributions on each subspace. Note that this radial distribution is equivalent to a Gamma distribution whose variables have been raised to the power of $\frac{1}{p_\emptyset}$. In the following we will denote distributions of this type with $\gamma_p(u,s)$, where u and s are the shape and scale parameter of the Gamma distribution, respectively. The particular γ_p distribution that results in independent subspaces has arbitrary scale but shape parameter $u = \frac{n}{p_\emptyset}$. When using any other radial distribution, the different subspaces do not factorize, and the distribution is also not an ISA model. In that sense L_p-nested symmetric distributions are a generalization of ISA. Note, however, that not every ISA model is also L_p-nested symmetric since not every product of arbitrary distributions on the subspaces, even if they are L_p-spherically symmetric, must also be L_p-nested.

It is natural to ask, whether L_p-nested symmetric distributions can serve as a prior distribution $p(\boldsymbol{y}|\boldsymbol{\vartheta})$ over hidden factors in over-complete linear models of the form

$$p(\boldsymbol{x}|W,\sigma,\boldsymbol{\vartheta}) = \int p(\boldsymbol{x}|W\boldsymbol{y},\sigma)p(\boldsymbol{y}|\boldsymbol{\vartheta})d\boldsymbol{y},$$

where $p(\boldsymbol{x}|W\boldsymbol{y})$ represents the likelihood of the observed data point $\boldsymbol{x}$ given the hidden factors $\boldsymbol{y}$ and the over-complete matrix W. For example, $p(\boldsymbol{x}|W\boldsymbol{y},\sigma) = \mathcal{N}(W\boldsymbol{y},\sigma\cdot I)$ could be a Gaussian like in Olshausen and Field (1996). Unfortunately, such a model would suffer from the same problems as all over-complete linear models: While sampling from the prior is straightforward sampling from the posterior $p(\boldsymbol{y}|\boldsymbol{x},W,\boldsymbol{\vartheta},\sigma)$ is difficult because a whole subspace of $\boldsymbol{y}$ leads to the same $\boldsymbol{x}$.

Since parameter estimation either involves solving the high-dimensional integral $p(\boldsymbol{x}|W,\sigma,\boldsymbol{\vartheta}) = \int p(\boldsymbol{x}|W\boldsymbol{y},\sigma)p(\boldsymbol{y}|\boldsymbol{\vartheta})d\boldsymbol{y}$ or sampling from the posterior, learning is computationally demanding in such models. Various methods have been proposed to learn W, ranging from sampling the posterior only at its maximum (Olshausen and Field, 1996), approximating the posterior with a Gaussian via the Laplace approximation (Lewicki and Olshausen, 1999) or using Expectation Propagation (Seeger, 2008). In particular, all of the above studies either do not fit hyper-parameters $\boldsymbol{\vartheta}$ for the prior (Olshausen and Field, 1996; Lewicki and Olshausen, 1999) or rely on the factorial structure of it (Seeger, 2008). Since L_p-nested symmetric distributions do not provide such a factorial prior, Expectation Propagation is not directly applicable. An approximation like in Lewicki and Olshausen (1999) might be possible, but additionally estimating the parameters $\boldsymbol{\vartheta}$ of the L_p-nested symmetric distribution adds another level of complexity in the estimation procedure. Exploring such over-complete linear models with a non-factorial prior may be an interesting direction to investigate, but it will need a significant amount of additional numerical and algorithmical work to find an efficient and robust estimation procedure.

9. Nested Radial Factorization with L_p-Nested Symmetric Distributions

L_p-nested symmetric distribution also give rise to a non-linear ICA algorithm for linearly mixed non-factorial L_p-nested hidden sources $\boldsymbol{y}$. The idea is similar to the radial factorization algorithms proposed by Lyu and Simoncelli (2009) and Sinz and Bethge (2009). For this reason, we call it *nested radial factorization (NRF)*. For a one layer L_p-nested tree, NRF is equivalent to radial factorization as described in Sinz and Bethge (2009). If additionally p is set to $p = 2$, one obtains the radial Gaussianization by Lyu and Simoncelli (2009). Therefore, NRF is a generalization of radial Factorization. It has been demonstrated that radial factorization algorithms outperform linear ICA on natural image patches (Lyu and Simoncelli, 2009; Sinz and Bethge, 2009). Since L_p-nested symmetric distributions are slightly better in likelihood on natural image patches (Sinz et al., 2009b) and since the difference in the average log-likelihood directly corresponds to the reduction in dependencies between the single variables (Sinz and Bethge, 2009), NRF will slightly outperform radial factorization on natural images. For other types of data the performance will depend on how well the hidden sources can be modeled by a linear superposition of—possibly non-independent—L_p-nested symmetrically distributed sources. Here we state the algorithm as a possible application of L_p-nested symmetric distributions for unsupervised learning.

The idea is based on the observation that the choice of the radial distribution ϕ already determines the type of L_p-nested symmetric distribution. This also means that by changing the radial distribution by remapping the data, the distribution could possibly be turned in a factorial one. Radial factorization algorithms fit an L_p-spherically symmetric distribution with a very flexible radial distribution to the data and map this radial distribution ϕ_s (s for source) into the one of a p-generalized Normal distribution by the mapping

$$\boldsymbol{y} \mapsto \frac{(\mathcal{F}_{\perp\!\!\perp}^{-1} \circ \mathcal{F}_s)(\|\boldsymbol{y}\|_p)}{\|\boldsymbol{y}\|_p} \cdot \boldsymbol{y}, \tag{12}$$

where $\mathcal{F}_{\perp\!\!\perp}$ and $\mathcal{F}_s$ are the cumulative distribution functions of the two radial distributions involved. The mapping basically normalizes the demixed source $\boldsymbol{y}$ and rescales it with a new radius that has the correct distribution.

Exactly the same method cannot work for L_p-nested symmetric distributions since Proposition 9 states that there is no factorial distribution into which we could map the data by merely changing the radial distribution. Instead we have to remap the data in an iterative fashion beginning with changing the radial distribution at the root node into the radial distribution of the L_p-nested ISA shown in Equation (11). Once the nodes are independent, we repeat this procedure for each of the child nodes independently, then for their child nodes and so on, until only leaves are left. The rescaling of the radii is a non-linear mapping since the transform in Equation (12) is non-linear. Therefore, NRF is a non-linear ICA algorithm.

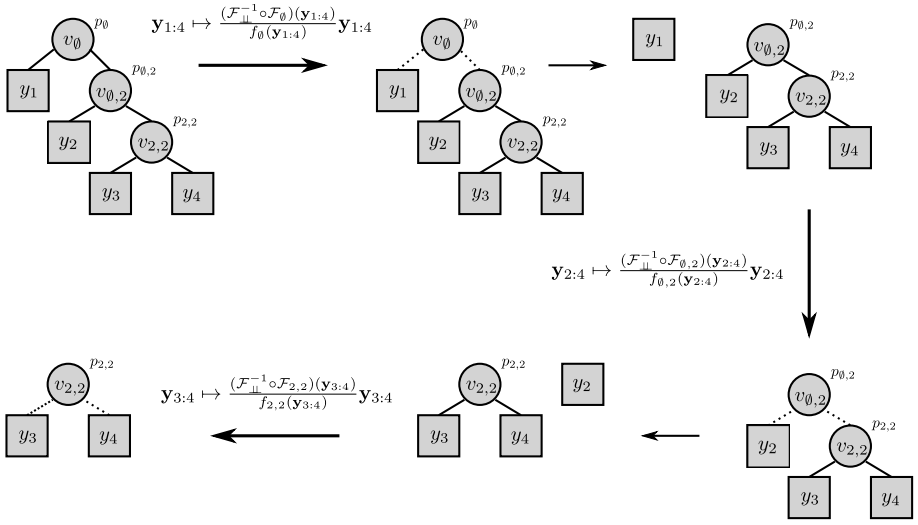


Figure 6: L_p-nested non-linear ICA for the tree of Example 6: For an arbitrary L_p-nested symmetric distribution, using Equation (12), the radial distribution can be remapped such that the children of the root node become independent. This is indicated in the plot via dotted lines. Once the data have been rescaled with that mapping, the children of root node can be separated. The remaining subtrees are again L_p-nested symmetric and have a particular radial distribution that can be remapped into the same one that makes their root nodes' children independent. This procedure is repeated until only leaves are left.

We demonstrate this with a simple example.

Example 6 *Consider the function*

$$f(\mathbf{y}) = \left(|y_1|^{p_\emptyset} + \left(|y_2|^{p_{\emptyset,2}} + \left(|y_3|^{p_{2,2}} + |y_4|^{p_{2,2}} \right)^{\frac{p_{\emptyset,2}}{p_{2,2}}} \right)^{\frac{p_\emptyset}{p_{\emptyset,2}}} \right)^{\frac{1}{p_\emptyset}}$$

for $\mathbf{y} = W\boldsymbol{x}$ where W has been estimated by fitting an L_p-nested symmetric distribution with a flexible radial distribution to $W\boldsymbol{x}$ as described in Section 5. Assume that the data has already been transformed once with the mapping of Equation (12)*. This means that the current radial distribution*

is given by (11) where we chose $s = 1$ for convenience. This yields a distribution of the form

$$\rho(\mathbf{y}) = \frac{p_\emptyset}{\Gamma\left[\frac{n}{p_\emptyset}\right]} \exp\left(-|y_1|^{p_\emptyset} - \left(|y_2|^{p_{\emptyset,2}} + \left(|y_3|^{p_{2,2}} + |y_4|^{p_{2,2}}\right)^{\frac{p_{\emptyset,2}}{p_{2,2}}}\right)^{\frac{p_\emptyset}{p_{\emptyset,2}}}\right)$$
$$\times \frac{1}{2^n} \prod_{I \in I} p_I^{\ell_I - 1} \frac{\Gamma\left[\frac{n_I}{p_I}\right]}{\prod_{k=1}^{\ell_I} \Gamma\left[\frac{n_{I,k}}{p_I}\right]}.$$

Now we can separate the distribution of y_1 from the distribution over $y_2, ..., y_4$. The distribution of y_1 is a p-generalized Normal

$$p(y_1) = \frac{p_\emptyset}{2\Gamma\left[\frac{1}{p_\emptyset}\right]} \exp\left(-|y_1|^{p_\emptyset}\right).$$

Thus the distribution of $y_2, ..., y_4$ is given by

$$\rho(y_2, ..., y_4) = \frac{p_\emptyset}{\Gamma\left[\frac{n_{\emptyset,2}}{p_\emptyset}\right]} \exp\left(-\left(|y_2|^{p_{\emptyset,2}} + \left(|y_3|^{p_{2,2}} + |y_4|^{p_{2,2}}\right)^{\frac{p_{\emptyset,2}}{p_{2,2}}}\right)^{\frac{p_\emptyset}{p_{\emptyset,2}}}\right)$$
$$\times \frac{1}{2^{n-1}} \prod_{I \in I \setminus \emptyset} p_I^{\ell_I - 1} \frac{\Gamma\left[\frac{n_I}{p_I}\right]}{\prod_{k=1}^{\ell_I} \Gamma\left[\frac{n_{I,k}}{p_I}\right]}.$$

By using Equation (9) we can identify the new radial distribution to be

$$\phi(v_{\emptyset,2}) = \frac{p_\emptyset v_{\emptyset,2}^{n-2}}{\Gamma\left[\frac{n_{\emptyset,2}}{p_\emptyset}\right]} \exp\left(-v_{\emptyset,2}^{p_\emptyset}\right).$$

Replacing this distribution by the one for the p-generalized Normal (for data we would use the mapping in Equation (12)), we obtain

$$\rho(y_2, ..., y_4) = \frac{p_{\emptyset,2}}{\Gamma\left[\frac{n_{\emptyset,2}}{p_{\emptyset,2}}\right]} \exp\left(-|y_2|^{p_{\emptyset,2}} - \left(|y_3|^{p_{2,2}} + |y_4|^{p_{2,2}}\right)^{\frac{p_{\emptyset,2}}{p_{2,2}}}\right)$$
$$\times \frac{1}{2^{n-1}} \prod_{I \in I \setminus \emptyset} p_I^{\ell_I - 1} \frac{\Gamma\left[\frac{n_I}{p_I}\right]}{\prod_{k=1}^{\ell_I} \Gamma\left[\frac{n_{I,k}}{p_I}\right]}.$$

Now, we can separate out the distribution of y_2 which is again p-generalized Normal. This leaves us with the distribution for y_3 and y_4

$$\rho(y_3, y_4) = \frac{p_{\emptyset,2}}{\Gamma\left[\frac{n_{2,2}}{p_{\emptyset,2}}\right]} \exp\left(-\left(|y_3|^{p_{2,2}} + |y_4|^{p_{2,2}}\right)^{\frac{p_{\emptyset,2}}{p_{2,2}}}\right) \frac{1}{2^{n-2}} \prod_{I \in I \setminus \{\emptyset,(\emptyset,2)\}} p_I^{\ell_I - 1} \frac{\Gamma\left[\frac{n_I}{p_I}\right]}{\prod_{k=1}^{\ell_I} \Gamma\left[\frac{n_{I,k}}{p_I}\right]}.$$

For this distribution we can repeat the same procedure which will also yield p-generalized Normal distributions for y_3 and y_4.

Algorithm 2 Recursion NRF($\boldsymbol{y}, f, \phi_s$)

Input: Data point $\boldsymbol{y}$, L_p-nested function f, current radial distribution ϕ_s,
Output: Non-linearly transformed data point $\boldsymbol{y}$
Algorithm

1. Set the target radial distribution to be $\phi_{\perp\!\!\!\perp} \leftarrow \gamma_p \left(\frac{n_\emptyset}{p_\emptyset}, \frac{\Gamma\left[\frac{1}{p_\emptyset}\right]^{\frac{p_\emptyset}{2}}}{\Gamma\left[\frac{3}{p_\emptyset}\right]^{\frac{p_\emptyset}{2}}} \right)$
2. Set $\boldsymbol{y} \leftarrow \frac{\mathcal{F}_{\perp\!\!\!\perp}^{-1}(\mathcal{F}_s(f(\boldsymbol{y})))}{f(\boldsymbol{y})} \cdot \boldsymbol{y}$ where $\mathcal{F}$ denotes the cumulative distribution function of the respective ϕ.
3. For all children i of the root node that are not leaves:
   (a) Set $\phi_s \leftarrow \gamma_p \left(\frac{n_{\emptyset,i}}{p_\emptyset}, \frac{\Gamma\left[\frac{1}{p_\emptyset}\right]^{\frac{p_\emptyset}{2}}}{\Gamma\left[\frac{3}{p_\emptyset}\right]^{\frac{p_\emptyset}{2}}} \right)$
   (b) Set $\boldsymbol{y}_{\emptyset,i} \leftarrow \text{NRF}(\boldsymbol{y}_{\emptyset,i}, f_{\emptyset,i}, \phi_s)$. Note that in the recursion $\emptyset, i$ will become the new $\emptyset$.
4. Return $\boldsymbol{y}$

This non-linear procedure naturally carries over to arbitrary L_p-nested trees and distributions, thus yielding a general non-linear ICA algorithm for linearly mixed non-factorial L_p-nested symmetric sources. For generalizing Example 6, note the particular form of the radial distributions involved. As already noted above, the distribution (11) on the root node's values that makes its children statistically independent is that of a Gamma distributed variable with shape parameter $\frac{n_\emptyset}{p_\emptyset}$ and scale parameter s which has been raised to the power of $\frac{1}{p_\emptyset}$. In Section 8 we denoted this class of distributions with $\gamma_p[u,s]$, where u and s are the shape and the scale parameter, respectively. Interestingly, the radial distributions of the root node's children are also γ_p except that the shape parameter is $\frac{n_{\emptyset,i}}{p_\emptyset}$. The goal of the radial remapping of the children's values is hence just changing the shape parameter from $\frac{n_{\emptyset,i}}{p_\emptyset}$ to $\frac{n_{\emptyset,i}}{p_{\emptyset,i}}$. Of course, it is also possible to change the scale parameter of the single distributions during the radial remappings. This will not affect the statistical independence of the resulting variables. In the general algorithm, that we describe now, we choose s such that the transformed data is white.

The algorithm starts with fitting a general L_p-nested model of the form $\rho(W\boldsymbol{x})$ as described in Section 5. Once this is done, the linear demixing matrix W is fixed and the hidden non-factorial sources are recovered via $\boldsymbol{y} = W\boldsymbol{x}$. Afterwards, the sources $\boldsymbol{y}$ are non-linearly made independent by calling the recursion specified in Algorithm 2 with the parameters $W\boldsymbol{x}$, f and ϕ, where ϕ is the radial distribution of the estimated model.

The computational complexity for transforming a single data point is $O(n^2)$ because of the matrix multiplication $W\boldsymbol{x}$. In the non-linear transformation, each single data dimension is not rescaled more that n times which means that the rescaling is certainly also in $O(n^2)$.

An important aspect of NRF is that it yields a probabilistic model for the transformed data. This model is simply a product of n independent exponential power marginals. Since the radial remappings do not change the likelihood, the likelihood of the non-linearly separated data is the

same as the likelihood of the data under L_p-nested symmetric distribution that was fitted to it in the first place. However, in some cases, one might like to fit a different distribution to the outcome of Algorithm 2. In that case the determinant of the transformation is necessary to determine the likelihood of the input data—and not the transformed one—under the model. The following lemma provides the determinant of the Jacobian for the non-linear rescaling.

Lemma 11 (Determinant of the Jacobian) *Let $\boldsymbol{z} = NRF(W\boldsymbol{x}, f, \phi_s)$ as described above. Let $\boldsymbol{t}_I$ denote the values of $W\boldsymbol{x}$ below the inner node I which have been transformed with Algorithm 2 up to node I. Let $g_I(r) = (\mathcal{F}_{\phi_{\perp\!\!\!\perp}} \circ \mathcal{F}_{\phi_s})(r)$ denote the radial transform at node I in Algorithm 2. Furthermore, let $\mathcal{I}$ denote the set of all inner nodes, excluding the leaves. Then, the determinant of the Jacobian $\left(\frac{\partial z_i}{\partial x_j}\right)_{ij}$ is given by*

$$\left|\det \frac{\partial z_i}{\partial x_j}\right| = |\det W| \cdot \prod_{I \in \mathcal{I}} \left| \frac{g_I(f_I(\boldsymbol{t}_I))^{n_I - 1}}{f_I(\boldsymbol{t}_I)^{n_I - 1}} \cdot \frac{\phi_s(f_I(\boldsymbol{t}_I))}{\phi_{\perp\!\!\!\perp}(g_I(f_I(\boldsymbol{t}_I)))} \right|$$

Proof The proof can be found in the Appendix E. ■

10. Conclusion

In this article we presented a formal treatment of the first tractable subclass of ν-spherical distributions which generalizes the important family of L_p-spherically symmetric distributions. We derived an analytical expression for the normalization constant, introduced a coordinate system particularly tailored to L_p-nested functions, and computed the determinant of the Jacobian for the corresponding coordinate transformation. Using these results, we introduced the uniform distribution on the L_p-nested unit sphere and the general form of an L_p-nested symmetric distribution for arbitrary L_p-nested functions and radial distributions. We also derived an expression for the joint distribution of inner nodes of an L_p-nested tree and derived a sampling scheme for an arbitrary L_p-nested symmetric distribution.

L_p-nested symmetric distributions naturally provide the class of probability distributions corresponding to mixed norm priors, allowing full Bayesian inference in the corresponding probabilistic models. We showed that a robustness result for Bayesian inference of the location parameter known for L_p-spherically symmetric distributions carries over to the L_p-nested symmetric class. We discussed the relationship of L_p-nested symmetric distributions to indepedent component (ICA) and independent subspace Analysis (ISA), as well as its applicability as a prior distribution in overcomplete linear models. Finally, we showed how L_p-nested symmetric distributions can be used to construct a non-linear ICA algorithm called nested radial factorization (NRF).

The application of L_p-nested symmetric distribution has been presented in a previous conference paper (Sinz et al., 2009b). Code for training this class of distribution is provided online under `http://www.kyb.tuebingen.mpg.de/bethge/code/`.

Acknowledgments

We would like to thank Eero Simoncelli for bringing up the problem whether the class of L_p-spherical distributions can be generalized to L_p-nested symmetric distributions. Furthermore, we

want to thank Sebastian Gerwinn, Suvrit Sra, Reshad Hosseini, Lucas Theis, Holly Gerhard, and Sina Tootoonian for fruitful discussions and feedback on the manuscript. Finally, we would like to thank the anonymous reviewers for their comments that helped to improve the manuscript.

This work is supported by the German Ministry of Education, Science, Research and Technology through the Bernstein prize to MB (BMBF; FKZ: 01GQ0601), a scholarship to FS by the German National Academic Foundation, and the Max Planck Society.

Appendix A. Determinant of the Jacobian

Proof [Lemma 2] The proof is very similar to the one in Song and Gupta (1997). To derive Equation (2) one needs to expand the Jacobian of the inverse coordinate transformation with respect to the last column using the Laplace's expansion of the determinant. The term Δ_n can be factored out of the determinant and cancels due to the absolute value around it. Therefore, the determinant of the coordinate transformation does not depend on Δ_n.

The partial derivatives of the inverse coordinate transformation are given by:

$$\begin{aligned}
\frac{\partial}{\partial u_k} x_i &= \delta_{ik} r \text{ for } 1 \leq i,k \leq n-1 \\
\frac{\partial}{\partial u_k} x_n &= \Delta_n r \frac{\partial u_n}{\partial u_k} \text{ for } 1 \leq k \leq n-1 \\
\frac{\partial}{\partial r} x_i &= u_i \text{ for } 1 \leq i \leq n-1 \\
\frac{\partial}{\partial r} x_n &= \Delta_n u_n.
\end{aligned}$$

Therefore, the structure of the Jacobian is given by

$$\mathcal{J} = \begin{pmatrix} r & \dots & 0 & u_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & r & u_{n-1} \\ \Delta_n r \frac{\partial u_n}{\partial u_1} & \dots & \Delta_n r \frac{\partial u_n}{\partial u_{n-1}} & \Delta_n u_n \end{pmatrix}.$$

Since we are only interested in the absolute value of the determinant and since $\Delta_n \in \{-1, 1\}$, we can factor out Δ_n and drop it. Furthermore, we can factor out r from the first $n-1$ columns which yields

$$|\det \mathcal{J}| = r^{n-1} \left| \det \begin{pmatrix} 1 & \dots & 0 & u_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 1 & u_{n-1} \\ \frac{\partial u_n}{\partial u_1} & \dots & \frac{\partial u_n}{\partial u_{n-1}} & u_n \end{pmatrix} \right|.$$

Now we can use the Laplace's expansion of the determinant with respect to the last column. For that purpose, let $\mathcal{J}_i$ denote the matrix which is obtained by deleting the last column and the ith row

from $\mathcal{J}$. This matrix has the following structure

$$\mathcal{J}_i = \begin{pmatrix} 1 & & & 0 & & & \\ & \ddots & & 0 & & & \\ & & 1 & 0 & & & \\ & & & \vdots & 1 & & \\ & & & 0 & & \ddots & \\ & & & 0 & & & 1 \\ \frac{\partial u_n}{\partial u_1} & & & \frac{\partial u_n}{\partial u_i} & & & \frac{\partial u_n}{\partial u_{n-1}} \end{pmatrix}.$$

We can transform $\mathcal{J}_i$ into a lower triangular matrix by moving the column with all zeros and $\frac{\partial u_n}{\partial u_i}$ bottom entry to the rightmost column of $\mathcal{J}_i$. Each swapping of two columns introduces a factor of -1. In the end, we can compute the value of $\det \mathcal{J}_i$ by simply taking the product of the diagonal entries and obtain $\det \mathcal{J}_i = (-1)^{n-1-i} \frac{\partial u_n}{\partial u_i}$. This yields

$$\begin{aligned} |\det \mathcal{J}| &= r^{n-1} \left(\sum_{k=1}^{n} (-1)^{n+k} u_k \det \mathcal{J}_k \right) \\ &= r^{n-1} \left(\sum_{k=1}^{n-1} (-1)^{n+k} u_k \det \mathcal{J}_k + (-1)^{2n} \frac{\partial x_n}{\partial r} \right) \\ &= r^{n-1} \left(\sum_{k=1}^{n-1} (-1)^{n+k} u_k (-1)^{n-1-k} \frac{\partial u_n}{\partial u_k} + u_n \right) \\ &= r^{n-1} \left(-\sum_{k=1}^{n-1} u_k \frac{\partial u_n}{\partial u_k} + u_n \right). \end{aligned}$$

■

Before proving Proposition 3 stating that the determinant only depends on the terms $G_I(\boldsymbol{u}_{\widehat{I}})$ produced by the chain rule when used upwards in the tree, let us quickly outline the essential mechanism when taking the chain rule for $\frac{\partial u_n}{\partial u_q}$: Consider the tree corresponding to f. By definition u_n is the rightmost leaf of the tree. Let L, ℓ_L be the multi-index of u_n. As in the example, the chain rule starts at the leaf u_n and ascends in the tree until it reaches the lowest node whose subtree contains both, u_n and u_q. At this point, it starts descending the tree until it reaches the leaf u_q. Depending on whether the chain rule ascends or descends, two different forms of derivatives occur: while ascending, the chain rule produces $G_I(\boldsymbol{u}_{\widehat{I}})$-terms like the one in the example above. At descending, it produces $F_I(\boldsymbol{u}_I)$-terms. The general definitions of the $G_I(\boldsymbol{u}_{\widehat{I}})$- and $F_I(\boldsymbol{u}_I)$-terms are given by the recursive formulae

$$G_{I,\ell_I}(\boldsymbol{u}_{\widehat{I,\ell_I}}) = g_{I,\ell_I}(\boldsymbol{u}_{\widehat{I,\ell_I}})^{p_{I,\ell_I}-p_I} = \left(g_I(\boldsymbol{u}_{\widehat{I}})^{p_I} - \sum_{j=1}^{\ell_I-1} f_{I,j}(\boldsymbol{u}_{I,j})^{p_I} \right)^{\frac{p_{I,\ell_I}-p_I}{p_I}}$$

and

$$F_{I,i_r}(\boldsymbol{u}_{I,i_r}) = f_{I,i_r}(\boldsymbol{u}_{I,i_r})^{p_I - p_{I,i_r}} = \left(\sum_{k=1}^{\ell_{I,i_r}} f_{I,i_r,k}(\boldsymbol{u}_{I,i_r,k})^{p_{I,i_r}} \right)^{\frac{p_I - p_{I,i_r}}{p_{I,i_r}}}.$$

The next two lemmata are required for the proof of Proposition 3. We use the somewhat sloppy notation $k \in I, i_r$ if the variable u_k is a leaf in the subtree below I, i_r. The same notation is used for $\widehat{I}$.

Lemma 12 *Let $I = i_1, ..., i_{r-1}$ and I, i_r be any node of the tree associated with an L_p-nested function f. Then the following recursions hold for the derivatives of $g_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}})^{p_{I,i_r}}$ and $f_{I,i_r}^{p_I}(\boldsymbol{u}_{I,i_r})$ w.r.t u_q: If u_q is not in the subtree under the node I, i_r, that is, $k \notin I, i_r$, then*

$$\frac{\partial}{\partial u_q} f_{I,i_r}(\boldsymbol{u}_{I,i_r})^{p_I} = 0$$

and

$$\frac{\partial}{\partial u_q} g_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}})^{p_{I,i_r}} = \frac{p_{I,i_r}}{p_I} G_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}}) \cdot \begin{cases} \frac{\partial}{\partial u_q} g_I(\boldsymbol{u}_{\widehat{I}})^{p_I} & \text{if } q \in I \\ -\frac{\partial}{\partial u_q} f_{I,j}(\boldsymbol{u}_{I,j})^{p_I} & \text{if } q \in I, j \end{cases}$$

for $q \in I, j$ and $q \notin I, k$ for $k \neq j$. Otherwise

$$\frac{\partial}{\partial u_q} g_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}})^{p_{I,i_r}} = 0 \text{ and } \frac{\partial}{\partial u_q} f_{I,i_r}(\boldsymbol{u}_{I,i_r})^{p_I} = \frac{p_I}{p_{I,i_r}} F_{I,i_r}(\boldsymbol{u}_{I,i_r}) \frac{\partial}{\partial u_q} f_{I,i_r,s}(\boldsymbol{u}_{I,i_r,s})^{p_{I,i_r}}$$

for $q \in I, i_r, s$ and $q \notin I, i_r, k$ for $k \neq s$.

Proof Both of the first equations are obvious, since only those nodes have a non-zero derivative for which the subtree actually depends on u_q. The second equations can be seen by direct computation

$$\begin{aligned}
\frac{\partial}{\partial u_q} g_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}})^{p_{I,i_r}} &= p_{I,i_r} g_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}})^{p_{I,i_r}-1} \frac{\partial}{\partial u_q} G_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}}) \\
&= p_{I,i_r} g_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}})^{p_{I,i_r}-1} \frac{\partial}{\partial u_q} \left(g_I(\boldsymbol{u}_{\widehat{I}})^{p_I} - \sum_{j=1}^{\ell_I - 1} f_{I,j}(\boldsymbol{u}_{I,j})^{p_I} \right)^{\frac{1}{p_I}} \\
&= \frac{p_{I,i_r}}{p_I} g_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}})^{p_{I,i_r}-1} g_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}})^{1-p_I} \frac{\partial}{\partial u_q} \left(g_I(\boldsymbol{u}_{\widehat{I}})^{p_I} - \sum_{j=1}^{\ell_I - 1} f_{I,j}(\boldsymbol{u}_{I,j})^{p_I} \right) \\
&= \frac{p_{I,i_r}}{p_I} G_{I,i_r}(\boldsymbol{u}_{\widehat{I,i_r}}) \cdot \begin{cases} \frac{\partial}{\partial u_q} g_I(\boldsymbol{u}_{\widehat{I}})^{p_I} & \text{if } q \in I \\ -\frac{\partial}{\partial u_q} f_{I,j}(\boldsymbol{u}_{I,j})^{p_I} & \text{if } q \in I, j \end{cases}
\end{aligned}$$

Similarly

$$\begin{aligned}
\frac{\partial}{\partial u_q} f_{I,i_r}(\boldsymbol{u}_{I,i_r})^{p_I} &= p_I f_{I,i_r}(\boldsymbol{u}_{I,i_r})^{p_I-1} \frac{\partial}{\partial u_q} f_{I,i_r}(\boldsymbol{u}_{I,i_r}) \\
&= p_I f_{I,i_r}(\boldsymbol{u}_{I,i_r})^{p_I-1} \frac{\partial}{\partial u_q} \left(\sum_{k=1}^{\ell_{I,i_r}} f_{I,i_r,k}(\boldsymbol{u}_{I,i_r,k})^{p_{I,i_r}} \right)^{\frac{1}{p_{I,i_r}}} \\
&= \frac{p_I}{p_{I,i_r}} f_{I,i_r}(\boldsymbol{u}_{I,i_r})^{p_I-1} f_{I,i_r}(\boldsymbol{u}_{I,i_r})^{1-p_{I,i_r}} \frac{\partial}{\partial u_q} f_{I,i_r,s}(\boldsymbol{u}_{I,i_r,s})^{p_{I,i_r}} \\
&= \frac{p_I}{p_{I,i_r}} F_{I,i_r}(\boldsymbol{u}_{I,i_r}) \frac{\partial}{\partial u_q} f_{I,i_r,s}(\boldsymbol{u}_{I,i_r,s})^{p_{I,i_r}}
\end{aligned}$$

for $k \in I, i_r, s$. ■

The next lemma states the form of the whole derivative $\frac{\partial u_n}{\partial u_q}$ in terms of the $G_I(\boldsymbol{u}_{\widehat{I}})$- and $F_I(\boldsymbol{u}_I)$-terms.

Lemma 13 *Let $|u_q| = v_{\ell_1,\ldots,\ell_m,i_1,\ldots,i_t}$, $|u_n| = v_{\ell_1,\ldots,\ell_d}$ with $m < d$. The derivative of u_n w.r.t. u_q is given by*

$$\begin{aligned}
\frac{\partial}{\partial u_q} u_n = &-G_{\ell_1,\ldots,\ell_d}(\boldsymbol{u}_{\widehat{\ell_1,\ldots,\ell_d}}) \cdot \ldots \cdot G_{\ell_1,\ldots,\ell_{m+1}}(\boldsymbol{u}_{\ell_1,\ldots,\widehat{\ell_{m+1}}}) \\
&\times F_{\ell_1,\ldots,\ell_m,i_1}(\boldsymbol{u}_{\ell_1,\ldots,\ell_m,i_1}) \cdot F_{\ell_1,\ldots,\ell_m,i_1,\ldots,i_{t-1}}(\boldsymbol{u}_{\ell_1,\ldots,\ell_m,i_1,\ldots,i_{t-1}}) \cdot \Delta_q |u_q|^{p_{\ell_1,\ldots,\ell_m,i_1,\ldots,i_{t-1}}-1}
\end{aligned}$$

with $\Delta_q = \operatorname{sgn} u_q$ and $|u_q|^p = (\Delta_q u_q)^p$. In particular

$$\begin{aligned}
u_q \frac{\partial}{\partial u_q} u_n = &-G_{\ell_1,\ldots,\ell_d}(\boldsymbol{u}_{\widehat{\ell_1,\ldots,\ell_d}}) \cdot \ldots \cdot G_{\ell_1,\ldots,\ell_{m+1}}(\boldsymbol{u}_{\ell_1,\ldots,\widehat{\ell_{m+1}}}) \\
&\times F_{\ell_1,\ldots,\ell_m,i_1}(\boldsymbol{u}_1) \cdot F_{\ell_1,\ldots,\ell_m,i_1,\ldots,i_{t-1}}(\boldsymbol{u}_{\ell_1,\ldots,\ell_m,i_1}) \cdot |u_q|^{p_{\ell_1,\ldots,\ell_m,i_1,\ldots,i_{t-1}}}.
\end{aligned}$$

Proof Successive application of Lemma (12). ■

Proof [Proposition 3] Before we begin with the proof, note that $F_I(\boldsymbol{u}_I)$ and $G_I(\boldsymbol{u}_{\widehat{I}})$ fulfill following equalities

$$\begin{aligned}
G_{I,i_m}(\boldsymbol{u}_{\widehat{I,i_m}})^{-1} g_{I,i_m}(\boldsymbol{u}_{\widehat{I,i_m}})^{p_{I,i_m}} &= g_{I,i_m}(\boldsymbol{u}_{\widehat{I,i_m}})^{p_I} \\
&= g_I(\boldsymbol{u}_{\widehat{I}})^{p_I} - \sum_{k=1}^{\ell_I-1} F_{I,k}(\boldsymbol{u}_{I,k}) f_{I,k}(\boldsymbol{u}_{I,k})^{p_{I,k}} \qquad (13)
\end{aligned}$$

and

$$f_{I,i_m}(\boldsymbol{u}_{I,i_m})^{p_{I,i_m}} = \sum_{k=1}^{\ell_{I,i_m}} F_{I,i_m,k}(\boldsymbol{u}_{I,i_m,k}) f_{I,i_m,k}(\boldsymbol{u}_{I,i_m,k})^{p_{I,i_m,k}}. \qquad (14)$$

Now let $L = \ell_1,\ldots,\ell_{d-1}$ be the multi-index of the parent of u_n. We compute $\frac{1}{r^{n-1}}|\det \mathcal{J}|$ and obtain the result by solving for $|\det \mathcal{J}|$. As shown in Lemma (2) $\frac{1}{r^{n-1}}|\det \mathcal{J}|$ has the form

$$\frac{1}{r^{n-1}}|\det \mathcal{J}| = -\sum_{k=1}^{n-1} \frac{\partial u_n}{\partial u_k} \cdot u_k + u_n.$$

By definition $u_n = g_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}}) = g_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{p_{L,\ell_d}}$. Now, assume that $u_m, ..., u_{n-1}$ are children of L, that is, $u_k = v_{L,I,i_t}$ for some $I, i_t = i_1, ..., i_t$ and $m \leq k < n$. Remember, that by Lemma (13) the terms $u_q \frac{\partial}{\partial u_q} u_n$ for $m \leq q < n$ have the form

$$u_q \frac{\partial}{\partial u_q} u_n = - G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}}) \cdot F_{L,i_1}(\boldsymbol{u}_{L,i_1}) \cdot ... \cdot F_{L,I}(\boldsymbol{u}_{L,I}) \cdot |u_q|^{p_{\ell_1,...,\ell_{d-1},i_1,...,i_{t-1}}}.$$

Using Equation (13), we can expand the determinant as follows

$$\begin{aligned}
&- \sum_{k=1}^{n-1} \frac{\partial u_n}{\partial u_k} \cdot u_k + g_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{p_{L,\ell_d}} \\
= &- \sum_{k=1}^{m-1} \frac{\partial u_n}{\partial u_k} \cdot u_k - \sum_{k=m}^{n-1} \frac{\partial u_n}{\partial u_k} \cdot u_k + g_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{p_{L,\ell_d}} \\
= &- \sum_{k=1}^{m-1} \frac{\partial u_n}{\partial u_k} \cdot u_k \\
&+ G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}}) \left(- \sum_{k=m}^{n-1} G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{-1} \frac{\partial u_n}{\partial u_k} \cdot u_k + G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{-1} g_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{p_{L,\ell_d}} \right) \\
= &- \sum_{k=1}^{m-1} \frac{\partial u_n}{\partial u_k} \cdot u_k \\
&+ G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}}) \left(- \sum_{k=m}^{n-1} G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{-1} \frac{\partial u_n}{\partial u_k} \cdot u_k + g_L(\boldsymbol{u}_{\widehat{L}})^{p_L} - \sum_{k=1}^{\ell_d - 1} F_{L,k}(\boldsymbol{u}_{L,k}) f_{L,k}(\boldsymbol{u}_{L,k})^{p_{L,k}} \right).
\end{aligned}$$

Note that all terms $G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{-1} \frac{\partial u_n}{\partial u_k} \cdot u_k$ for $m \leq k < n$ now have the form

$$G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{-1} u_k \frac{\partial}{\partial u_k} u_n = -F_{L,i_1}(\boldsymbol{u}_{L,i_1}) \cdot ... \cdot F_{L,I}(\boldsymbol{u}_{L,I}) \cdot |u_q|^{p_{\ell_1,...,\ell_{d-1},i_1,...,i_{t-1}}}$$

since we constructed them to be neighbors of u_n. However, with Equation (14), we can further expand the sum $\sum_{k=1}^{\ell_d - 1} F_{L,k}(\boldsymbol{u}_{L,k}) f_{L,k}(\boldsymbol{u}_{L,k})^{p_{L,k}}$ down to the leaves $u_m, ..., u_{n-1}$. When doing so we end up with the same factors $F_{L,i_1}(\boldsymbol{u}_{L,i_1}) \cdot ... \cdot F_{L,I}(\boldsymbol{u}_{L,I}) \cdot |u_q|^{p_{\ell_1,...,\ell_{d-1},i_1,...,i_{t-1}}}$ as in the derivatives $G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{-1} u_q \frac{\partial}{\partial u_q} u_n$. This means exactly that

$$- \sum_{k=m}^{n-1} G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{-1} \frac{\partial u_n}{\partial u_k} \cdot u_k = \sum_{k=1}^{\ell_d - 1} F_{L,k}(\boldsymbol{u}_{L,k}) f_{L,k}(\boldsymbol{u}_{L,k})^{p_{L,k}}$$

and, therefore,

$$
\begin{aligned}
&-\sum_{k=1}^{m-1}\frac{\partial u_n}{\partial u_k}\cdot u_k \\
&+G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})\left(-\sum_{k=m}^{n-1}G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})^{-1}\frac{\partial u_n}{\partial u_k}\cdot u_k+g_L(\boldsymbol{u}_{\widehat{L}})^{p_L}-\sum_{k=1}^{\ell_d-1}F_{L,k}(\boldsymbol{u}_{L,k})f_{L,k}(\boldsymbol{u}_{L,k})^{p_{L,k}}\right)\\
=&-\sum_{k=1}^{m-1}\frac{\partial u_n}{\partial u_k}\cdot u_k \\
&+G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})\left(\sum_{k=1}^{\ell_d-1}F_{L,k}(\boldsymbol{u}_{L,k})f_{L,k}(\boldsymbol{u}_{L,k})^{p_{L,k}}+g_L(\boldsymbol{u}_{\widehat{L}})^{p_L}-\sum_{k=1}^{\ell_d-1}F_{L,k}(\boldsymbol{u}_{L,k})f_{L,k}(\boldsymbol{u}_{L,k})^{p_{L,k}}\right)\\
=&-\sum_{k=1}^{m-1}\frac{\partial u_n}{\partial u_k}\cdot u_k+G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})g_L(\boldsymbol{u}_{\widehat{L}})^{p_L}.
\end{aligned}
$$

By factoring out $G_{L,\ell_d}(\boldsymbol{u}_{\widehat{L,\ell_d}})$ from the equation, the terms $\frac{\partial u_n}{\partial u_k}\cdot u_k$ loose the G_{L,ℓ_d} in front and we get basically the same equation as before, only that the new leaf (the new "u_n") is $g_L(\boldsymbol{u}_{\widehat{L}})^{p_L}$ and we got rid of all the children of L. By repeating that procedure up to the root node, we successively factor out all $G_{L'}(\boldsymbol{u}_{\widehat{L'}})$ for $L'\in\mathcal{L}$ until all terms of the sum vanish and we are only left with $v_\emptyset=1$. Therefore, the determinant is

$$
\frac{1}{r^{n-1}}|\det \mathcal{J}|=\prod_{L\in\mathcal{L}}G_L(\boldsymbol{u}_{\widehat{L}})
$$

which completes the proof. ■

Appendix B. Volume and Surface of the L_p-Nested Unit Sphere

Proof [Proposition 4] We obtain the volume by computing the integral $\int_{f(\boldsymbol{x})\le R}d\boldsymbol{x}$. Differentiation with respect to R yields the surface area. For symmetry reasons we can compute the volume only on the positive quadrant $\mathbb{R}^n_+$ and multiply the result with 2^n later to obtain the full volume and surface area. The strategy for computing the volume is as follows. We start with inner nodes I that are parents of leaves only. The value v_I of such a node is simply the L_{p_I} norm of its children. Therefore, we can convert the integral over the children of I with the transformation of Gupta and Song (1997). This maps the leaves $\boldsymbol{v}_{I,1:\ell_I}$ into v_I and "angular" variables $\tilde{\boldsymbol{u}}$. Since integral borders of the original integral depend only on the value of v_I and not on $\tilde{\boldsymbol{u}}$, we can separate the variables $\tilde{\boldsymbol{u}}$ from the radial variables v_I and integrate the variables $\tilde{\boldsymbol{u}}$ separately. The integration over $\tilde{\boldsymbol{u}}$ yields a certain factor, while the variable v_I effectively becomes a new leaf.

Now suppose I is the parent of leaves only. Without loss of generality let the ℓ_I leaves correspond to the last ℓ_I coefficients of $\boldsymbol{x}$. Let $\boldsymbol{x} \in \mathbb{R}^n_+$. Carrying out the first transformation and integration yields

$$\begin{aligned}\int_{f(\boldsymbol{x})\leq R} d\boldsymbol{x} &= \int_{f(\boldsymbol{x}_{1:n-\ell_I}, v_I)\leq R} \int_{\tilde{\boldsymbol{u}}\in\mathcal{V}_+^{\ell_I-1}} v_I^{\ell_I-1} \left(1-\sum_{i=1}^{\ell_I-1}\tilde{u}_i^{p_I}\right)^{\frac{1-p_I}{p_I}} dv_I d\tilde{\boldsymbol{u}} d\boldsymbol{x}_{1:n-\ell_I} \\ &= \int_{f(\boldsymbol{x}_{1:n-\ell_I}, v_I)\leq R} v_I^{n_I-1} dv_I d\boldsymbol{x}_{1:n-\ell_I} \times \int_{\tilde{\boldsymbol{u}}\in\mathcal{V}_+^{\ell_I-1}} \left(1-\sum_{i=1}^{\ell_I-1}\tilde{u}_i^{p_I}\right)^{\frac{n_{I,\ell_I}-p_I}{p_I}} d\tilde{\boldsymbol{u}}.\end{aligned}$$

where $\mathcal{V}_+$ denotes the intersection of the positive quadrant and the L_{p_I}-norm unit ball. For solving the second integral we make the pointwise transformation $s_i = \tilde{u}_i^{p_I}$ and obtain

$$\begin{aligned}\int_{\tilde{\boldsymbol{u}}\in\mathcal{V}_+^{\ell_I-1}} \left(1-\sum_{i=1}^{\ell_I-1}\tilde{u}_i^{p_I}\right)^{\frac{n_{I,\ell_I}-p_I}{p_I}} d\tilde{\boldsymbol{u}} &= \frac{1}{p_I^{\ell_I-1}} \int_{\sum s_i\leq 1} \left(1-\sum_{i=1}^{\ell_I-1} s_i\right)^{\frac{n_{I,\ell_I}}{p_I}-1} \prod_{i=1}^{\ell_I-1} s_i^{\frac{1}{p_I}-1} d\boldsymbol{s}_{\ell_I-1} \\ &= \frac{1}{p_I^{\ell_I-1}} \prod_{k=1}^{\ell_I-1} B\left[\frac{\sum_{i=1}^k n_{I,k}}{p_I}, \frac{n_{I,k+1}}{p_I}\right] \\ &= \frac{1}{p_I^{\ell_I-1}} \prod_{k=1}^{\ell_I-1} B\left[\frac{k}{p_I}, \frac{1}{p_I}\right]\end{aligned}$$

by using the fact that the transformed integral has the form of an unnormalized Dirichlet distribution and, therefore, the value of the integral must equal its normalization constant.

Now, we solve the integral

$$\int_{f(\boldsymbol{x}_{1:n-\ell_I}, v_I)\leq R} v_I^{n_I-1} dv_I d\boldsymbol{x}_{1:n-\ell_I}. \tag{15}$$

We carry this out in exactly the same manner as we solved the previous integral. We need only to make sure that we only contract nodes that have only leaves as children (remember that radii of contracted nodes become leaves) and we need to find a formula describing how the factors $v_I^{n_I-1}$ propagate through the tree.

For the latter, we first state the formula and then prove it via induction. For notational convenience let $\mathcal{J}$ denote the set of multi-indices corresponding to the contracted leaves, $\boldsymbol{x}_{\hat{\mathcal{J}}}$ the remaining coefficients of $\boldsymbol{x}$ and $\boldsymbol{v}_{\mathcal{J}}$ the vector of leaves resulting from contraction. The integral which is left to solve after integrating over all $\tilde{\boldsymbol{u}}$ is given by (remember that n_J denotes real leaves, that is, the ones corresponding to coefficients of $\boldsymbol{x}$):

$$\int_{f(\boldsymbol{x}_{\hat{\mathcal{J}}}, \boldsymbol{v}_{\mathcal{J}})\leq R} \prod_{J\in\mathcal{J}} v_J^{n_J-1} d\boldsymbol{v}_{\mathcal{J}} d\boldsymbol{x}_{\hat{\mathcal{J}}}.$$

We already proved the first induction step by computing Equation (15). For computing the general induction step suppose I is an inner node whose children are leaves or contracted leaves. Let $\mathcal{J}'$ be the set of contracted leaves under I and $\mathcal{K} = \mathcal{J}\backslash\mathcal{J}'$. Transforming the children of I into radial

coordinates by Gupta and Song (1997) yields

$$\begin{aligned}
\int_{f(\boldsymbol{x}_{\widehat{\mathcal{J}}},\boldsymbol{v}_{\mathcal{J}})\le R}\prod_{J\in\mathcal{J}} v_J^{n_J-1}d\boldsymbol{v}_{\mathcal{J}}d\boldsymbol{x}_{\widehat{\mathcal{J}}} &= \int_{f(\boldsymbol{x}_{\widehat{\mathcal{J}}},\boldsymbol{v}_{\mathcal{J}})\le R}\left(\prod_{K\in\mathcal{K}} v_K^{n_K-1}\right)\cdot\left(\prod_{J'\in\mathcal{J}'} v_{J'}^{n_{J'}-1}\right)d\boldsymbol{v}_{\mathcal{J}}d\boldsymbol{x}_{\widehat{\mathcal{J}}}\\
&= \int_{f(\boldsymbol{x}_{\widehat{\mathcal{K}}},\boldsymbol{v}_{\mathcal{K}},v_I)\le R}\int_{\tilde{\boldsymbol{u}}_{\ell_I-1}\in\mathcal{V}_+^{\ell_I-1}}\left(\left(1-\sum_{i=1}^{\ell_I-1}\tilde{u}_i^{p_I}\right)^{\frac{1-p_I}{p_I}} v_I^{\ell_I-1}\right)\cdot\left(\prod_{K\in\mathcal{K}} v_K^{n_K-1}\right)\\
&\times\left(\left(v_I\left(1-\sum_{i=1}^{\ell_I-1}\tilde{u}_i^{p_I}\right)^{\frac{1}{p_I}}\right)^{n_{\ell_I}-1}\prod_{k=1}^{\ell_I-1}(v_I\tilde{u}_k)^{n_k-1}\right)d\boldsymbol{x}_{\widehat{\mathcal{K}}}d\boldsymbol{v}_{\mathcal{K}}dv_Id\tilde{\boldsymbol{u}}_{\ell_I-1}\\
&= \int_{f(\boldsymbol{x}_{\widehat{\mathcal{K}}},\boldsymbol{v}_{\mathcal{K}},v_I)\le R}\int_{\tilde{\boldsymbol{u}}_{\ell_I-1}\in\mathcal{V}_+^{\ell_I-1}}\left(\prod_{K\in\mathcal{K}} v_K^{n_K-1}\right)\\
&\times\left(v_I^{\ell_I-1+\sum_{i=1}^{\ell_I}(n_i-1)}\left(1-\sum_{i=1}^{\ell_I-1}\tilde{u}_i^{p_I}\right)^{\frac{n_{\ell_I}-p_I}{p_I}}\prod_{k=1}^{\ell_I-1}\tilde{u}_k^{n_k-1}\right)d\boldsymbol{x}_{\widehat{\mathcal{K}}}d\boldsymbol{v}_{\mathcal{K}}dv_Id\tilde{\boldsymbol{u}}_{\ell_I-1}\\
&= \int_{f(\boldsymbol{x}_{\widehat{\mathcal{K}}},\boldsymbol{v}_{\mathcal{K}},v_I)\le R}\left(\prod_{K\in\mathcal{K}} v_K^{n_K-1}\right)v_I^{n_I-1}d\boldsymbol{x}_{\widehat{\mathcal{K}}}d\boldsymbol{v}_{\mathcal{K}}dv_I\\
&\times\int_{\tilde{\boldsymbol{u}}_{\ell_I-1}\in\mathcal{V}_+^{\ell_I-1}}\left(1-\sum_{i=1}^{\ell_I-1}\tilde{u}_i^{p_I}\right)^{\frac{n_{\ell_I}-p_I}{p_I}}\prod_{k=1}^{\ell_I-1}\tilde{u}_k^{n_k-1}d\tilde{\boldsymbol{u}}_{\ell_I-1}.
\end{aligned}$$

Again, by transforming it into a Dirichlet distribution, the latter integral has the solution

$$\int_{\tilde{\boldsymbol{u}}_{\ell_I-1}\in\mathcal{V}_+^{\ell_I-1}}\left(1-\sum_{i=1}^{\ell_I-1}\tilde{u}_i^{p_I}\right)^{\frac{n_{\ell_I}-p_I}{p_I}}\prod_{k=1}^{\ell_I-1}\tilde{u}_k^{n_k-1}d\tilde{\boldsymbol{u}}_{\ell_I-1}=\prod_{k=1}^{\ell_I-1}B\left[\frac{\sum_{i=1}^{k}n_{I,k}}{p_I},\frac{n_{I,k+1}}{p_I}\right]$$

while the remaining former integral has the form

$$\int_{f(\boldsymbol{x}_{\widehat{\mathcal{K}}},\boldsymbol{v}_{\mathcal{K}},v_I)\le R}\left(\prod_{K\in\mathcal{K}} v_K^{n_K-1}\right)v_I^{n_I-1}d\boldsymbol{x}_{\widehat{\mathcal{K}}}d\boldsymbol{v}_{\mathcal{K}}dv_I=\int_{f(\boldsymbol{x}_{\widehat{\mathcal{J}}},\boldsymbol{v}_{\mathcal{J}})\le R}\prod_{J\in\mathcal{J}} v_J^{n_J-1}d\boldsymbol{v}_{\mathcal{J}}d\boldsymbol{x}_{\widehat{\mathcal{J}}}$$

as claimed.

By carrying out the integration up to the root node, the remaining integral becomes

$$\int_{v_\emptyset\le R} v_\emptyset^{n-1}dv_\emptyset=\int_0^R v_\emptyset^{n-1}dv_\emptyset=\frac{R^n}{n}.$$

Collecting the factors from integration over the $\tilde{\boldsymbol{u}}$ proves the Equations (5) and (7). Using $B[a,b]=\frac{\Gamma[a]\Gamma[b]}{\Gamma[a+b]}$ yields Equations (6) and (8). ■

Appendix C. Layer Marginals

Proof [Proposition 7]

$$\begin{aligned}\rho(\boldsymbol{x}) &= \frac{\phi(f(\boldsymbol{x}))}{\mathcal{S}_f(f(\boldsymbol{x}))} \\ &= \frac{\phi(f(\boldsymbol{x}_{1:n-\ell_I}, v_I, \tilde{\boldsymbol{u}}_{\ell_I-1}, \Delta_n))}{\mathcal{S}_f(f(\boldsymbol{x}))} \cdot v_I^{\ell_I-1} \left(1 - \sum_{i=1}^{\ell_I-1} |\tilde{u}_i|^{p_I}\right)^{\frac{1-p_I}{p_I}}\end{aligned}$$

where $\Delta_n = \operatorname{sign}(x_n)$. Note that f is invariant to the actual value of Δ_n. However, when integrating it out, it yields a factor of 2. Integrating out $\tilde{\boldsymbol{u}}_{\ell_I-1}$ and Δ_n now yields

$$\begin{aligned}\rho(\boldsymbol{x}_{1:n-\ell_I}, v_I) &= \frac{\phi(f(\boldsymbol{x}_{1:n-\ell_I}, v_I))}{\mathcal{S}_f(f(\boldsymbol{x}))} \cdot v_I^{\ell_I-1} \frac{2^{\ell_I}\Gamma^{\ell_I}\left[\frac{1}{p_I}\right]}{p_I^{\ell_I-1}\Gamma\left[\frac{\ell_I}{p_I}\right]} \\ &= \frac{\phi(f(\boldsymbol{x}_{1:n-\ell_I}, v_I))}{\mathcal{S}_f(f(\boldsymbol{x}_{1:n-\ell_I}, v_I))} \cdot v_I^{\ell_I-1}\end{aligned}$$

Now, we can go on and integrate out more subtrees. For that purpose, let $\boldsymbol{x}_{\hat{\mathcal{J}}}$ denote the remaining coefficients of $\boldsymbol{x}$, $\boldsymbol{v}_{\mathcal{J}}$ the vector of leaves resulting from the kind of contraction just shown for v_I, and $\mathcal{J}$ the set of multi-indices corresponding to the "new leaves", that is, node v_I after contraction. We obtain the following equation

$$\rho(\boldsymbol{x}_{\hat{\mathcal{J}}}, \boldsymbol{v}_{\mathcal{J}}) = \frac{\phi(f(\boldsymbol{x}_{\hat{\mathcal{J}}}, \boldsymbol{v}_{\mathcal{J}}))}{S_f(f(\boldsymbol{x}_{\hat{\mathcal{J}}}, \boldsymbol{v}_{\mathcal{J}}))} \prod_{J \in \mathcal{J}} v_J^{n_J-1}.$$

where n_J denotes the number of leaves in the subtree under the node J. The calculations for the proof are basically the same as the one for proposition (4). ■

Appendix D. Factorial L_p-Nested Distributions

Proof [Proposition 9] Since the single x_i are independent, $f_1(\boldsymbol{x}_1), ..., f_{\ell_\emptyset}(\boldsymbol{x}_{\ell_\emptyset})$ and, therefore, $v_1, ..., v_{\ell_\emptyset}$ must be independent as well ($\boldsymbol{x}_i$ are the elements of $\boldsymbol{x}$ in the subtree below the ith child of the root node). Using Corollary 8 we can write the density of $v_1, ..., v_{\ell_\emptyset}$ as (the function name g is unrelated to the usage of the function g above)

$$\rho(\boldsymbol{v}_{1:\ell_\emptyset}) = \prod_{i=1}^{\ell_\emptyset} h_i(v_i) = g(\|\boldsymbol{v}_{1:\ell_\emptyset}\|_{p_\emptyset}) \prod_{i=1}^{\ell_\emptyset} v_i^{n_i-1}$$

with

$$g(\|\boldsymbol{v}_{1:\ell_\emptyset}\|_{p_\emptyset}) = \frac{p_\emptyset^{\ell_\emptyset-1}\Gamma\left[\frac{n}{p_\emptyset}\right]}{\|\boldsymbol{v}_{1:\ell_\emptyset}\|_{p_\emptyset}^{n-1} 2^m \prod_{k=1}^{\ell_\emptyset} \Gamma\left[\frac{n_k}{p_\emptyset}\right]} \phi(\|\boldsymbol{v}_{1:\ell_\emptyset}\|_{p_\emptyset})$$

Since the integral over g is finite, it follows from Sinz et al. (2009a) that g has the form $g(\|\boldsymbol{v}_{1:\ell_\emptyset}\|_{p_\emptyset}) = \exp(a_\emptyset\|\boldsymbol{v}_{1:\ell_\emptyset}\|_{p_\emptyset}^{p_\emptyset} + b_\emptyset)$ for appropriate constants $a_\emptyset$ and $b_\emptyset$. Therefore, the marginals have the form

$$h_i(v_i) = \exp(a_\emptyset v_i^{p_\emptyset} + c_\emptyset) v_i^{n_i-1}. \tag{16}$$

On the other hand, the particular form of g implies that the radial density has the form $\phi(f(\boldsymbol{x})) \propto f(\boldsymbol{x})^{(n-1)} \exp(a_\emptyset f(\boldsymbol{x})^{p_\emptyset} + b_\emptyset)^{p_\emptyset}$. In particular, this implies that the root node's children $f_i(\boldsymbol{x}_i)$ $(i = 1,...,\ell_\emptyset)$ are independent and L_p-nested symmetric again. With the same argument as above, it follows that their children $\boldsymbol{v}_{i,1:\ell_i}$ follow the distribution $\rho(v_{i,1},...,v_{i,\ell_i}) = \exp(a_i\|\boldsymbol{v}_{i,1:\ell_i}\|_{p_i}^{p_i} + b_i)\prod_{j=1}^{\ell_i} v_{i,j}^{n_{i,j}-1}$. Transforming that distribution to L_p-spherically symmetric polar coordinates $v_i = \|\boldsymbol{v}_{i,1:\ell_i}\|_{p_i}$ and $\tilde{\boldsymbol{u}} = \boldsymbol{v}_{i,1:\ell_i-1}/\|\boldsymbol{v}_{i,1:\ell_i}\|_{p_i}$ as in Gupta and Song (1997), we obtain the form

$$\begin{aligned}
\rho(v_i, \tilde{\boldsymbol{u}}) &= \exp(a_i v_i^{p_i} + b_i) v_i^{\ell_i-1} \left(1 - \sum_{j=1}^{\ell_i-1} |\tilde{u}_j|^{p_i}\right)^{\frac{1-p_i}{p_i}} \left(v_i \left(1 - \sum_{j=1}^{\ell_i-1} |\tilde{u}_j|^{p_i}\right)^{\frac{1}{p_i}}\right)^{n_{i,\ell_i}-1} \prod_{j=1}^{\ell_i-1} (\tilde{u}_j v_i)^{n_{i,j}-1} \\
&= \exp(a_i v_i^{p_i} + b_i) v_i^{n_i-1} \left(1 - \sum_{j=1}^{\ell_i-1} |\tilde{u}_j|^{p_i}\right)^{\frac{n_{i,\ell_i}-p_i}{p_i}} \prod_{j=1}^{\ell_i-1} \tilde{u}_j^{n_{i,j}-1},
\end{aligned}$$

where the second equation follows the same calculations as in the proof of Proposition 4. After integrating out $\tilde{\boldsymbol{u}}$, assuming that the x_i are statistically independent, we obtain the density of v_i which is equal to (16) if and only if $p_i = p_\emptyset$. However, if $p_\emptyset$ and p_i are equal, the hierarchy of the L_p-nested function shrinks by one layer since p_i and $p_\emptyset$ cancel themselves. Repeated application of the above argument collapses the complete L_p-nested tree until one effectively obtains an L_p-spherical function. Since the only factorial L_p-spherically symmetric distribution is the p-generalized Normal (Sinz et al., 2009a) the claim follows. ■

Appendix E. Determinant of the Jacobian for NRF

Proof [Lemma 11] The proof is a generalization of the proof of Lyu and Simoncelli (2009). Due to the chain rule the Jacobian of the entire transformation is the multiplication of the Jacobians for each single step, that is, the rescaling of a subset of the dimensions for one single inner node. The Jacobian for the other dimensions is simply the identity matrix. Therefore, the determinant of the Jacobian for each single step is the determinant for the radial transformation on the respective dimensions. We show how to compute the determinant for a single step.

Assume that we reached a particular node I in Algorithm 2. The leaves, which have been rescaled by the preceding steps, are called $\boldsymbol{t}_I$. Let $\boldsymbol{\xi}_I = \frac{g_I(f_I(\boldsymbol{t}_I))}{f_I(\boldsymbol{t}_I))} \cdot \boldsymbol{t}_I$ with $g_I(r) = (\mathcal{F}_{\perp\!\!\!\perp}^{-1} \circ \mathcal{F}_s)(r)$. The general form of a single Jacobian is

$$\frac{\partial \boldsymbol{\xi}_I}{\partial \boldsymbol{t}_I} = \boldsymbol{t}_I \cdot \frac{\partial}{\partial \boldsymbol{t}_I}\left(\frac{g_I(f_I(\boldsymbol{t}_I))}{f_I(\boldsymbol{t}_I)}\right) + \frac{g_I(f_I(\boldsymbol{t}_I))}{f_I(\boldsymbol{t}_I)} I_{n_I},$$

where

$$\frac{\partial}{\partial \boldsymbol{t}_I}\left(\frac{g_I(f_I(\boldsymbol{t}_I))}{f_I(\boldsymbol{t}_I)}\right) = \left(\frac{g_I'(f_I(\boldsymbol{t}_I))}{f_I(\boldsymbol{t}_I)} - \frac{g_I(f_I(\boldsymbol{t}_I))}{f_I(\boldsymbol{t}_I)^2}\right) \frac{\partial}{\partial \boldsymbol{t}_I} f_I(\boldsymbol{t}_I).$$

Let y_i be a leave in the subtree under I and let $I, J_1, ..., J_k$ be the path of inner nodes from I to y_i, then

$$\frac{\partial}{\partial y_i} f_I(\boldsymbol{t}_I) = v_I^{1-p_I} v_{J_1}^{p_I - p_{J_1}} \cdot \ldots \cdot v_k^{p_{J_{k-1}} - p_{J_k}} |y_i|^{p_{J_k}-1} \cdot \operatorname{sgn} y_i.$$

If we denote $r = f_I(\boldsymbol{t}_I)$ and $\zeta_i = v_{J_1}^{p_I - p_{J_1}} \cdot \ldots \cdot v_k^{p_{J_{k-1}} - p_{J_k}} |y_i|^{p_{J_k}-1} \cdot \operatorname{sgn} y_i$ for the respective J_k, we obtain

$$\det\left(\boldsymbol{t}_I \cdot \frac{\partial}{\partial \boldsymbol{t}_I}\left(\frac{g_I(f_I(\boldsymbol{t}_I))}{f_I(\boldsymbol{t}_I)}\right) + \frac{g_I(f_I(\boldsymbol{t}_I))}{f_I(\boldsymbol{t}_I)} I_{n_I}\right) = \det\left(\left(g_I'(r) - \frac{g_I(r)}{r}\right) r^{-p_I} \boldsymbol{t}_I \cdot \boldsymbol{\zeta}^\top + \frac{g_I(r)}{r} I_{n_I}\right).$$

Now we can use Sylvester's determinant formula $\det(I_n + b\boldsymbol{t}_I\boldsymbol{\zeta}^\top) = \det(1 + b\boldsymbol{t}_I^\top\boldsymbol{\zeta}) = 1 + b\boldsymbol{t}_I^\top\boldsymbol{\zeta}$ or equivalently

$$\begin{aligned}
\det(aI_n + b\boldsymbol{t}_I\boldsymbol{\zeta}^\top) &= \det\left(a \cdot \left(I_n + \frac{b}{a}\boldsymbol{t}_I\boldsymbol{\zeta}^\top\right)\right) \\
&= a^n \det\left(I_n + \frac{b}{a}\boldsymbol{t}_I\boldsymbol{\zeta}^\top\right) \\
&= a^{n-1}(a + b\boldsymbol{t}_I^\top\boldsymbol{\zeta}),
\end{aligned}$$

as well as $\boldsymbol{t}_I^\top\boldsymbol{\zeta} = f_I(\boldsymbol{t}_I)^{p_I} = r^{p_I}$ to see that

$$\begin{aligned}
\det\left(\left(g_I'(r) - \frac{g_I(r)}{r}\right) r^{-p_I}\boldsymbol{t}_I \cdot \boldsymbol{\zeta}^\top + \frac{g_I(r)}{r} I_n\right) &= \frac{g_I(r)^{n-1}}{r^{n-1}} \det\left(\left(g_I'(r) - \frac{g_I(r)}{r}\right) r^{-p_I}\boldsymbol{t}_I^\top \cdot \boldsymbol{\zeta} + \frac{g_I(r)}{r}\right) \\
&= \frac{g_I(r)^{n-1}}{r^{n-1}} \det\left(g_I'(r) - \frac{g_I(r)}{r} + \frac{g_I(r)}{r}\right) \\
&= \frac{g_I(r)^{n-1}}{r^{n-1}} \frac{d}{dr} g_I(r).
\end{aligned}$$

$\frac{d}{dr} g_I(r)$ is readily computed via $\frac{d}{dr} g_I(r) = \frac{d}{dr}(\mathcal{F}_{\perp\perp}^{-1} \circ \mathcal{F}_s)(r) = \frac{\phi_s(r)}{\phi_{\perp\perp}(g_I(r))}$.

Multiplying the single determinants along with $\det W$ for the final step of the chain rule completes the proof. ■

References

P-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton Univ Pr, Dec 2007. ISBN 0691132984.

M. Bethge. Factorial coding of natural images: How effective are linear model in removing higher-order dependencies? *J. Opt. Soc. Am. A*, 23(6):1253–1268, June 2006.

A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, 1999. ISSN 0895-4798.

J. Eichhorn, F. Sinz, and M. Bethge. Natural image coding in v1: How much use is orientation selectivity? *PLoS Comput Biol*, 5(4), Apr 2009.

K. T. Fang, S. Kotz, and K. W. Ng. *Symmetric Multivariate and Related Distributions*. Chapman and Hall New York, 1990.

C. Fernandez, J. Osiewalski, and M.F.J. Steel. Modeling and inference with ν-spherical distributions. *Journal of the American Statistical Association*, 90(432):1331–1340, Dec 1995. URL http://www.jstor.org/stable/2291523.

A.K. Gupta and D. Song. L_p-norm spherical distribution. *Journal of Statistical Planning and Inference*, 60:241–260, 1997.

A.E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58(3):54–59, 1962.

A. Hyvärinen and P. Hoyer. Emergence of phase and shift invariant features by decomposition of natural images into independent feature subspaces. *Neural Comput.*, 12(7):1705–1720, 2000.

A. Hyvärinen and U. Köster. Fastisa: A fast fixed-point algorithm for independent subspace analysis. In *Proc. of ESANN*, pages 371–376, 2006.

A. Hyvärinen and U. Köster. Complex cell pooling and the statistics of natural images. *Network: Computation in Neural Systems*, 18(2):81–100, 2007.

A. Hyvärinen and Erkki O. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, Oct 1997. doi: 10.1162/neco.1997.9.7.1483.

D. Kelker. Distribution theory of spherical distributions and a location-scale parameter generalization. *Sankhya: The Indian Journal of Statistics, Series A*, 32(4):419–430, Dec 1970. doi: 10.2307/25049690. URL http://www.jstor.org/stable/25049690.

M. Kowalski, E. Vincent, and R. Gribonval. Under-determined source separation via mixed-norm regularized minimization. In *Proceedings of the European Signal Processing Conference*, 2008.

TW. Lee and M. Lewicki. The generalized gaussian mixture model using ica. In P. Pajunen and J. Karhunen, editors, *ICA' 00*, pages 239–244, Helsinki, Finland, june 2000.

M. S. Lewicki. Efficient coding of natural sounds. *Nat Neurosci*, 5(4):356–363, Apr 2002. doi: 10.1038/nn831.

M.S. Lewicki and B.A. Olshausen. Probabilistic framework for the adaptation and comparison of image codes. *J. Opt. Soc. Am. A*, 16:1587–1601, 1999.

S. Lyu and E. P. Simoncelli. Nonlinear extraction of independent components of natural images using radial gaussianization. *Neural Computation*, 21(6):1485–1519, Jun 2009. doi: 10.1162/neco.2009.04-08-773.

J. H. Manton. Optimization algorithms exploiting unitary constraints. *IEEE Transactions on Signal Processing*, 50:635 – 650, 2002.

B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:560–561, 1996.

J. Osiewalski and M. F. J. Steel. Robust bayesian inference in l_q-spherical models. *Biometrika*, 80 (2):456–460, Jun 1993. URL http://www.jstor.org/stable/2337215.

M. W. Seeger. Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9:759–813, 04 2008. URL http://www.jmlr.org/papers/volume9/seeger08a/seeger08a.pdf.

E.P. Simoncelli. Statistical models for images: compression, restoration and synthesis. In *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems & Computers, 1997.*, volume 1, pages 673–678 vol.1, 1997. doi: 10.1109/ACSSC.1997.680530.

F. Sinz and M. Bethge. The conjoint effect of divisive normalization and orientation selectivity on redundancy reduction. In D. Schuurmans Y. Bengio L. Bottou Koller, D., editor, *Twenty-Second Annual Conference on Neural Information Processing Systems*, pages 1521–1528, Red Hook, NY, USA, 06 2009. Curran. URL http://nips.cc/Conferences/2008/.

F. Sinz, S. Gerwinn, and M. Bethge. Characterization of the p-generalized normal distribution. *Journal of Multivariate Analysis*, 100(5):817–820, May 2009a. doi: 10.1016/j.jmva.2008.07.006.

F. Sinz, E. P. Simoncelli, and M. Bethge. Hierarchical modeling of local image features through L_p-nested symmetric distributions. In *Twenty-Third Annual Conference on Neural Information Processing Systems*, pages 1–9, 12 2009b. URL http://nips.cc/Conferences/2009/.

D. Song and A.K. Gupta. L_p-norm uniform distribution. *Proceedings of the American Mathematical Society*, 125:595–601, 1997.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 00359246. URL http://www.jstor.org/stable/2346178.

M.J. Wainwright and E.P. Simoncelli. Scale mixtures of Gaussians and the statistics of natural images. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Adv. Neural Information Processing Systems (NIPS*99)*, volume 12, pages 855–861, Cambridge, MA, May 2000. MIT Press.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B*, 68(1):49–67, 2006.

C. Zetzsche, B. Wegmann, and E. Barth. Nonlinear aspects of primary vision: entropy reduction beyond decorrelation. In *Int'l Symposium, Soc. for Information Display*, volume XXIV, pages 933–936. 1993.

L. Zhang, A. Cichocki, and S. Amari. Self-adaptive blind source separation based on activation functions adaptation. *Neural Networks, IEEE Transactions on*, 15:233–244, 2004.

P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, 2008.

Journal of Machine Learning Research 11 (2010) 3453-3479 Submitted 3/10; Revised 9/10; Published 11/10

Efficient Algorithms for Conditional Independence Inference

Remco Bouckaert[*] REMCO@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikato
Hamilton 3240, New Zealand

Raymond Hemmecke HEMMECKE@MA.TUM.DE
Silvia Lindner SLINDNER@MA.TUM.DE
Zentrum Mathematik
Technische Universtität Munich
Boltzmannstrasse 3
85747 Garching, Germany

Milan Studený STUDENY@UTIA.CAS.CZ
Institute of Information Theory and Automation of the ASCR
Pod Vodárenskou věží 4
18208 Prague, Czech Republic

Editor: Rina Dechter

Abstract

The topic of the paper is computer testing of (probabilistic) *conditional independence* (CI) implications by an algebraic method of structural imsets. The basic idea is to transform (sets of) CI statements into certain integral vectors and to verify by a computer the corresponding algebraic relation between the vectors, called the *independence implication*. We interpret the previous methods for computer testing of this implication from the point of view of polyhedral geometry. However, the main contribution of the paper is a new method, based on *linear programming* (LP). The new method overcomes the limitation of former methods to the number of involved variables. We recall/describe the theoretical basis for all four methods involved in our computational experiments, whose aim was to compare the efficiency of the algorithms. The experiments show that the LP method is clearly the fastest one. As an example of possible application of such algorithms we show that testing inclusion of Bayesian network structures or whether a CI statement is encoded in an acyclic directed graph can be done by the algebraic method.

Keywords: conditional independence inference, linear programming approach

1. Introduction

First, we explain the motivation and mention some previous work. Then we describe the aim and structure of the paper.

1.1 Motivation

Conditional independence (CI) is a highly important concept in statistics and artificial intelligence. Properties of probabilistic CI provide theoretical justification for the method of local computation

∗. Also at the University of Auckland, New Zealand.

(Cowel et al., 1999) which is at the core of probabilistic expert systems (Jensen, 2001), successfully applied in numerous areas. The importance of CI is given by its interpretation in terms of *relevance among symptoms* or variables in consideration (Pearl, 1988); that's why it is crucial in probabilistic reasoning. Traditional methods for describing (statistical models of) CI structures use graphs whose nodes correspond to variables in consideration; it leads to popular *graphical models* (Lauritzen, 1996).

Formal properties of probabilistic CI and the attempts to describe probabilistic *CI inference* in terms of mathematical logic have been traditional research topics since the late 1970's. Basic properties of CI, now known as the *semi-graphoid properties*, were accentuated in statistics by Dawid (1979). Pearl (1988) interpreted those properties, formulated in the form of simple implications between CI statements, as axioms for the relevance relation. Moreover, Pearl proposed to view graphs as "inference engines" devised for efficient representing and manipulating relevance relationships. His idea (see Pearl, 1988, page 14) was to use an *input list of CI statements* to construct a graph and then, by using a special graphical separation criterion, to read from the graph additional CI statements, implied by the input list through the axioms. The goal of such inference procedures is to enable one to determine, at any state of knowledge, what information is relevant to the task at hand and what can be ignored.

Pearl's intention led him to a conjecture that CI inference for discrete probabilistic distributions can be characterized by the semi-graphoid properties. The conjecture was refuted in Studený (1989); actually, it has been shown later that there is no finite system of properties of semi-graphoid type characterizing (discrete probabilistic) CI inference (Studený, 1992). Thus, the question of computer testing of CI inference became a topic of research interest.

In Studený (2005), the method of *structural imsets* has been proposed as a non-graphical algebraic device for describing probabilistic CI structures; its advantage over graphical approaches is that it allows one to describe any discrete probabilistic CI structure. The idea is to use, instead of graphs, certain special integral vectors (of high dimension), called *imsets*, as tools for describing CI structures and implementing the "inference engine" for CI implication. This is because the corresponding algebraic relation between structural imsets, called *independence implication*, gives a sufficient condition for (probabilistic) CI inference. The topic of this paper is computer testing of this implication. The intended use is

- computer testing of implications between CI statements, and
- checking equivalence of structural imsets by an algebraic method.

Another (indirect) source of motivation for this paper is learning *Bayesian network* (BN) structures by score and search methods (Neapolitan, 2004). The point is that every BN structure can be described uniquely by a simple algebraic representative, called the *standard imset*. It was shown in Studený (2005, Chapter 8) that every reasonable scoring function (for learning BN structures) can be viewed as an affine function of the standard imset. This observation opens the way to the application of efficient *linear programming* (LP) methods in this area (Studený, Vomlel, and Hemmecke, 2010). Nevertheless, if a graphical model of CI structure is described by an imset, a natural question arises: is there any criterion, a counterpart of the graphical separation criterion, which allows one to read from this algebraic representative all CI statements encoded in it? This question again leads to the task of (computer) testing independence implication.

1.2 Some Former Algorithms

Given structural imsets u and v (over a set of variables N), the intuitive meaning of the implication $u \rightharpoonup v$ ($\equiv$ u independence implies v) is that the CI structure induced by u contains the CI structure induced by v. In Studený (2005, § 6.2) two algebraic characterizations of the relation $u \rightharpoonup v$ were given. They established the theoretical basis for the first-generation algorithms for computer testing of $u \rightharpoonup v$. The limitation of these algorithms is that to implement them fully one needs some additional information obtainable as the result of computations, which were performed only for $|N| \leq 5$ (Studený, Bouckaert, and Kočka, 2000).

The theoretical aspects of their implementation were analyzed in Studený (2004), while their practical implementations were described in detail in Bouckaert and Studený (2007). Basically, there are two algorithms, which can be interpreted as mutually complementary procedures. One of them, based on so-called "direct" characterization of $u \rightharpoonup v$ (see Studený, 2005, § 6.2.1), is suitable to confirm the implication; that's why it was called the *verification algorithm* in Bouckaert and Studený (2007). The other algorithm, based on so-called "skeletal" characterization of $u \rightharpoonup v$ (see Studený, 2005, § 6.2.2), fits to disproving the implication; that's why it was named the *falsification algorithm*.

The main idea of Bouckaert and Studený (2007) was to combine both algorithms to get a more effective tool. Owing to the computations from Studený et al. (2000), the implemented versions of both these algorithms are guaranteed to give a decisive answer to any independence implication problem for $|N| \leq 5$. Nevertheless, the combined version has also been implemented for $|N| = 6$, although without a guaranteed response to each implication problem. Recently, the first-generation algorithms from Bouckaert and Studený (2007) have been applied (in a modified form) in connection with the lattice-theoretic approach to CI inference (Niepert, van Gucht, and Gyssens, 2008).

1.3 Aim of the Paper

In this paper, we bring a new view on the problem of testing CI inference. First, we interpret the previous methods from the point of view of polyhedral geometry. Second, this geometric interpretation and the *linear programming approach* lead to new methods and, consequently, to the second-generation algorithms for testing CI inference, which appear to be more efficient than the first-generation algorithms.

More specifically, the geometric view has recently helped to solve an open problem that was very closely related to the topic of CI inference (see Studený, 2005, Question 7). It was the question whether every structural imset is a *combinatorial imset*, that is, whether it can be written as the sum of elementary imsets (= imsets corresponding to elementary CI statements). The question has been answered negatively in Hemmecke et al. (2008), where an example of a structural imset over 5 variables was found that is not combinatorial.

This fact naturally leads to a more advanced question motivated by the topic of CI inference: what is the so-called *Hilbert basis* of the cone generated by standard imsets (cf., Studený, 2005, Theme 10). A recent achievement is that the Hilbert basis for $|N| = 5$ has been found as a result of computations by Bruns et al. (2010). This makes it possible to design two modifications of the verification algorithm for 5 variables. In fact, the CI inference problem is transformed to testing whether a given imset is combinatorial, respectively structural.

Another important idea brought by this paper is that every testing $u \rightharpoonup v$ task can be re-formulated as a special LP problem. Note that the LP approach was mentioned as a potential method for testing

this implication already in Studený (2004, § 5). However, in the present paper, we re-formulate this implication problem in a different way. The LP approach has the following advantages:

- it allows one to go far beyond the limit of 5 variables,
- the second-generation algorithms are much faster than the first-generation ones,
- the re-formulation makes it possible to apply highly effective software packages developed to solve LP problems.

The goal of this paper is to describe the idea of the new approach, the corresponding algorithms and the experiments, whose aim was to compare the new algorithms (with the old ones).

Moreover, to give an example of the possible use of the algorithms we prove a result related to learning Bayesian networks. We characterize the inclusion of BN structures in terms of their algebraic representatives, standard imsets. Given acyclic directed graphs G and H over N, we show that the BN structure induced by H is contained in the one induced by G iff the difference $u_G - u_H$ of their standard imsets is a combinatorial imset, which happens iff it is a structural imset. Thus, testing the inclusion can be transformed to testing combinatorial imsets (or structural ones). A consequence of this observation is an elegant algebraic procedure for reading CI statements represented in a standard imset u_G, a kind of counterpart of the graphical separation criterion. Since the standard imset u_G is quite simple we have reasons to conjecture that our procedure can be implemented with polynomial complexity with respect to $|N|$.

1.4 Structure of the Paper

Section 2 is devoted to the terminology for CI inference using structural imsets, while basic concepts from polyhedral geometry are recalled in Section 3. In Section 4, the methods we are using are explained and, in Section 5, the experiments we performed are described. In Conclusions we discuss the perspectives and formulate some open problems. Appendices contain some proofs, a table of types of the Hilbert basis elements for 5 variables, an illustrative example and a comment on possible interpretation of the LP method.

2. Concepts Concerning CI Inference

The symbol N will be used to denote a non-empty finite set of *variables* in consideration. Given disjoint sets of variables $A, B \subseteq N$, the juxtaposition AB will denote their union.

2.1 Conditional Independence

Let P be a discrete probability distribution over N specified by its density $p : \mathsf{X}_N \to [0,1]$, where $\mathsf{X}_N \equiv \prod_{i \in N} \mathsf{X}_i$ is the joint sample space, a product of non-empty finite individual sample spaces. Given $A \subseteq N$ and $x \in \mathsf{X}_N$, let x_A denote the respective marginal configuration of x and p_A the marginal density of p; by convention $p_\emptyset(-) = 1$. Given pairwise disjoint $A, B, C \subseteq N$, we say that A is *conditionally independent of B given C* with respect to P if

$$\forall x \in \mathsf{X}_N \quad p_{A \cup B \cup C}(x_{A \cup B \cup C}) \cdot p_C(x_C) = p_{A \cup C}(x_{A \cup C}) \cdot p_{B \cup C}(x_{B \cup C}) \,.$$

We write $A \perp\!\!\!\perp B\,|\,C\,[P]$ then and call it a *CI statement* with respect to P; if P is not material we just use the symbol $A \perp\!\!\!\perp B\,|\,C$. The *CI structure* induced by P consists of all CI statements valid with respect to P.

Now, let L be a set of CI statements, called an *input list*, and t a CI statement outside L. We say L *probabilistically implies* t if, for every discrete probability distribution P, whenever all statements in L are valid with respect to P then t is valid with respect to P, too. Classic examples of valid probabilistic CI implications are the *semi-graphoid properties* (Pearl, 1988):

Symmetry	$A \perp\!\!\!\perp B\,\|\,C$	$\Longrightarrow$	$B \perp\!\!\!\perp A\,\|\,C$,
Decomposition	$A \perp\!\!\!\perp BD\,\|\,C$	$\Longrightarrow$	$A \perp\!\!\!\perp D\,\|\,C$,
Weak union	$A \perp\!\!\!\perp BD\,\|\,C$	$\Longrightarrow$	$A \perp\!\!\!\perp B\,\|\,DC$,
Contraction	$A \perp\!\!\!\perp B\,\|\,DC$ & $A \perp\!\!\!\perp D\,\|\,C$	$\Longrightarrow$	$A \perp\!\!\!\perp BD\,\|\,C$.

A CI statement $A \perp\!\!\!\perp B\,|\,C$ is called *elementary* if both A and B are singletons and it is called *trivial* if either $A = \emptyset$ or $B = \emptyset$. Since trivial statements are always valid, it follows from the semi-graphoid properties that, for any discrete probability distribution P, the CI structure induced by P is uniquely determined by the list of elementary CI statements valid with respect to P.

2.2 Imsets

An *imset* over N is an integer-valued function on the power set of N, denoted by $\mathcal{P}(N)$ in the rest of the paper.[1] It can be viewed as a vector whose components, indexed by subsets of N, are integers. An easy example is the *zero imset*, denoted by 0; another simple example is the identifier δ_A of a subset $A \subseteq N$:

$$\delta_A(S) = \begin{cases} 1 & \text{if } S = A, \\ 0 & \text{if } S \subseteq N,\ S \neq A. \end{cases}$$

An imset associated with a CI statement $A \perp\!\!\!\perp B\,|\,C$ is then the combination

$$u_{\langle A,B|C\rangle} \equiv \delta_{ABC} + \delta_C - \delta_{AC} - \delta_{BC}\,.$$

The class of *elementary imsets* (over N), denoted by $\mathcal{E}(N)$, consists of imsets associated with elementary CI statements (over N).

A *combinatorial imset* is an imset u that can directly be decomposed into elementary imsets, that is,

$$u = \sum_{w \in \mathcal{E}(N)} k_w \cdot w \qquad \text{for some } k_w \in \mathbb{Z}_+\,.$$

We denote the class of combinatorial imsets over N by $\mathcal{C}(N)$.

An imset u over N will be called *structural* if there exists $n \in \mathbb{N}$ such that the multiple $n \cdot u$ is a combinatorial imset, that is,

$$n \cdot u = \sum_{w \in \mathcal{E}(N)} k_w \cdot w \quad \text{for some } n \in \mathbb{N},\ k_w \in \mathbb{Z}_+\,.$$

In other words, a structural imset is an imset which is a combination of elementary ones with non-negative rational coefficients. The class of structural imsets over N will be denoted by $\mathcal{S}(N)$; this

1. The word **imset** is an abbreviation for **i**nteger-valued **m**ulti**set**.

class of imsets is intended to describe CI structures. Clearly, $\mathcal{E}(N) \subseteq \mathcal{C}(N) \subseteq \mathcal{S}(N)$. One has $\mathcal{C}(N) = \mathcal{S}(N)$ for $|N| \leq 4$ (Studený, 1991), but $\mathcal{S}(N) \neq \mathcal{C}(N)$ for $|N| = 5$ (Hemmecke et al., 2008).

However, one can show (see Studený, 2005, Lemma 6.3) that there exists a smallest $n_* \in \mathbb{N}$, depending on $|N|$, such that $u \in \mathcal{S}(N) \Leftrightarrow n_* \cdot u \in \mathcal{C}(N)$ for every imset u over N. One has $n_* = 1$ for $|N| \leq 4$ (Studený, 1991) and one of the news brought by this paper is that $n_* = 2$ for $|N| = 5$ (see Proposition 6 in Appendix B). An open question is what is the exact value of n_* for every $|N|$ (cf., Studený, 2005, Theme 11).

2.3 Independence Implication

Let u, v be structural imsets over N. We say that u *independence implies* v and write $u \rightharpoonup v$ if there exists $k \in \mathbb{N}$ such that $k \cdot u - v$ is a structural imset:

$$u \rightharpoonup v \quad \text{iff} \quad \exists k \in \mathbb{N} \quad k \cdot u - v \in \mathcal{S}(N)\,. \tag{1}$$

Now, given $u \in \mathcal{S}(N)$ the corresponding CI structure $\mathcal{M}(u)$ consists of all CI statements $A \perp\!\!\!\perp B\,|\,C$ such that $u \rightharpoonup u_{\langle A,B|C\rangle}$. The importance of structural imsets follows from the fact that, for every discrete probability distribution P over N, there exists $u \in \mathcal{S}(N)$ such that the CI structure induced by P coincides with $\mathcal{M}(u)$ (use Studený, 2005, Theorem 5.2).

An equivalent characterization of independence implication is as follows. A real set function $m : \mathcal{P}(N) \to \mathbb{R}$ is called *supermodular* iff

$$m(C \cup D) + m(C \cap D) \geq m(C) + m(D) \qquad \text{for every } C, D \subseteq N.$$

Such a function can be viewed as a real vector whose components are indexed by subsets of N. The *scalar product* of m and an imset u over N is then

$$\langle m, u \rangle \equiv \sum_{S \subseteq N} m(S) \cdot u(S)\,.$$

The dual definition of independence implication is this (see Studený, 2005, Lemma 6.2): one has $u \rightharpoonup v$ iff, for every supermodular function m over N,

$$\langle m, u \rangle = 0 \quad \Longrightarrow \quad \langle m, v \rangle = 0\,. \tag{2}$$

The independence implication can be viewed as a sufficient condition for probabilistic CI implication. More specifically, given an input list L of CI statements, let us "translate" every statement s in L into the associated imset u_s and introduce a combinatorial imset $u_L \equiv \sum_{s \in L} u_s$. Then one has:

Proposition 1 Let L be an input list of CI statements and t another CI statement over N. If $u_L \rightharpoonup u_t$ then L probabilistically implies t.

The proof, based on observations from Studený (2005), is given in Appendix A. Note that the above-mentioned condition is a sufficient condition for the probabilistic CI implication, but not a necessary one. On the other hand, the analysis in the case of four variables suggests that it is quite good approximation of probabilistic implication. For $|N| = 4$, one has 22108 (formal) CI structures induced by structural imsets, while the actual number of (discrete) probabilistic CI structures is 18478 (Šimeček, 2007), and there are only about 20 types of probabilistic CI implications that are not derivable through Proposition 1.

The reader may be interested in whether there exists a limit for the factor k in (1). If we assume that the "input" u is a combinatorial imset and the "output" v an elementary one then such a limit exists. One can show (cf., Studený, 2005, Lemma 6.4) that the least $k_* \in \mathbb{N}$ exists such that

$$u \rightharpoonup v \quad \text{iff} \quad k_* \cdot u - v \in \mathcal{S}(N)\,.$$

It depends on $|N|$: $k_* = 1$ for $|N| \leq 4$ and $k_* = 7$ for $|N| = 5$ (Studený et al., 2000). It is an open question what is the exact value of k_* for $|N| \geq 6$ (cf., Studený, 2005, Theme 12).

2.4 Bayesian Network Translation

Bayesian networks (Pearl, 1988; Neapolitan, 2004) can be interpreted as graphical models of CI structures assigned to acyclic directed graphs. More specifically, given an acyclic directed graph G having N as the set of its nodes, the corresponding graphical separation criterion (for the definition see Lauritzen, 1996, § 3.2.2) defines the collection $I(G)$ of CI restrictions given by G. The corresponding statistical model then consists of probability distributions on X_N that are *Markovian with respect to* G, that is, satisfy those CI restrictions. Given two acyclic directed graphs G and H over N, we say they are *independence equivalent* if $I(G) = I(H)$; it essentially means they define the same statistical model.

Given an acyclic directed graph G over N, the *standard imset* for G, denoted by u_G, is given by the formula

$$u_G = \delta_N - \delta_\emptyset + \sum_{i \in N} \{\, \delta_{\mathrm{pa}_G(i)} - \delta_{\{i\} \cup \mathrm{pa}_G(i)} \,\}, \tag{3}$$

where $\mathrm{pa}_G(i) \equiv \{j \in N;\, j \to i \ \text{ in } G\}$ denotes the set of *parents* of a node i in G. Lemma 7.1 in Studený (2005) says that u_G is always a combinatorial imset and $\mathcal{M}(u_G) = I(G)$. Thus, the graphical separation criterion applied to G can be replaced by an algebraic criterion applied to u_G. Moreover, Corollary 7.1 in Studený (2005) adds that $u_G = u_H$ iff G and H are independence equivalent. That means, the standard imset is a unique representative of the equivalence class of graphs.

Note that u_G need not be the only combinatorial imset defining $I(G)$; it is the simplest such imset, a kind of "standard" representative. Using standard imsets we can easily characterize the inclusion for Bayesian networks:

Proposition 2 Let G and H be acyclic directed graphs over N. Then $I(H) \subseteq I(G)$ if and only if $u_G - u_H \in \mathcal{C}(N)$, which is also equivalent to $u_G - u_H \in \mathcal{S}(N)$.

Proof The equivalence $I(H) \subseteq I(G) \Leftrightarrow u_G - u_H \in \mathcal{C}(N)$ is proved in (Studený, 2005, Lemma 8.6). Clearly, $u_G - u_H \in \mathcal{C}(N)$ implies $u_G - u_H \in \mathcal{S}(N)$. Conversely, if $u_G - u_H \in \mathcal{S}(N)$ then, by (1), $u_G \rightharpoonup u_H$ and it makes no problem to show $\mathcal{M}(u_H) \subseteq \mathcal{M}(u_G)$. This means $I(H) \subseteq I(G)$. ■

We have some reasons to conjecture that testing whether a difference of two standard imsets is a combinatorial imset can be done with polynomial complexity in $|N|$; see Conclusions for the discussion. Moreover, let us emphasize that as far as we are aware of, there is no direct graphical characterization of independence inclusion; in our view, it is quite difficult task to describe this in graphical terms. Note that indirect transformational graphical description of independence inclusion by Chickering (2002) does not provide an algorithm for testing the inclusion for a given pair of

acyclic directed graphs G and H. Of course, a different situation is with testing independence equivalence, in which case a simple graphical characterization and algorithms are available.

Proposition 2 has the following consequence, which allows one to replace the graphical separation criterion by testing whether an imset is combinatorial.

Corollary 3 Given an acyclic directed graph G over N, the class $I(G)$ coincides with the collection of CI statements $A \perp\!\!\!\perp B \,|\, C$ such that $u_G - u_{\langle A,B|C\rangle} \in \mathcal{C}(N)$, respectively $u_G - u_{\langle A,B|C\rangle} \in \mathcal{S}(N)$.

Proof It is enough to construct, for every CI statements $A \perp\!\!\!\perp B \,|\, C$, an acyclic directed graph H such that $u_{\langle A,B|C\rangle} = u_H$. To this end, consider a total order of nodes in N in which C is followed A, B and $N \setminus ABC$, direct edges of the complete undirected graph over N according to this order and remove the arrows between A and B. ■

Of course, as mentioned above, there are elegant graphical separation criteria for testing whether a CI statement is represented in an acyclic directed graph. There are no doubts that these criteria can be implemented with polynomial complexity in $|N|$ (see Geiger, Verma, and Pearl, 1989) and that they are also appropriate for human users. However, our method proposed in Corollary 3 may appear to be suitable for computer testing, in particular in the situation when the Bayesian network structure is internally represented in a computer by the standard imset, as suggested in Studený, Vomlel, and Hemmecke (2010).

3. Basic Concepts from Polyhedral Geometry

Here we recall some well-known concepts and facts from the theory of convex polyhedra. Proofs can be found in textbooks on linear programming, for example Schrijver (1986).

Vectors are regarded as column vectors. Given two vectors $\mathbf{x}, \mathbf{y}$, their scalar product will be denoted either by $\langle \mathbf{x}, \mathbf{y} \rangle$ or, using matrix multiplication notation, by $\mathbf{x}^\intercal \mathbf{y}$, where $\mathbf{x}^\intercal$ is the transpose of $\mathbf{x}$. The inequality $\mathbf{x} \leq \mathbf{y}$ is meant in all components, and $\mathsf{0}$, respectively $\mathsf{1}$, is a vector all whose components are zeros, respectively ones.

3.1 Polyhedra, Polyhedral Cones and Farkas' Lemma

Systems of linear inequalities appear in many areas of mathematics. Let $\mathbf{A}$ be a $d \times n$-matrix and $\mathbf{b}$ a d-vector. The set of all solutions in $\mathbb{R}^n$ to the linear system $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ is called a *polyhedron*. If the defining matrix $\mathbf{A}$ and the right-hand side vector $\mathbf{b}$ are rational, the polyhedron is said to be *rational*. Polyhedra defined via homogeneous inequalities $\mathbf{A}\mathbf{x} \leq \mathsf{0}$ are called *polyhedral cones*.

A non-trivial fundamental result in polyhedral geometry states that every polyhedral cone C can equivalently be introduced as the *conical hull* $\mathrm{cone}(\mathsf{V})$ of finitely many points $\mathsf{V} \subseteq \mathbb{R}^n$, that is, the set of all finite conical combinations $\sum_t \lambda_t \cdot \mathbf{v}_t$ (with $\lambda_t \geq 0$ for all t) of elements $\mathbf{v}_t \in \mathsf{V}$ (see Schrijver, 1986, Corollary 7.1a). If the polyhedral cone is *rational* then it is the conical hull of finitely many *rational* points. A classic algorithmic task is to change between the *inner description* $\mathsf{C} = \mathrm{cone}(\mathsf{V})$ and the *outer description* $\mathsf{C} = \{\mathbf{x} \in \mathbb{R}^n; \mathbf{A}\mathbf{x} \leq \mathsf{0}\}$, and back. Standard software packages that allow one to switch between both descriptions are for example 4TI2 (4ti2 team, 2008), CDD (Fukuda, 2008), or CONVEX (Franz, 2009).

In our studies below, it will be important to decide whether a given polyhedron P is empty or not. Clearly, a certificate for $\mathsf{P} = \{\mathbf{x} \in \mathbb{R}^n; \mathbf{A}\mathbf{x} \leq \mathbf{b}\} \neq \emptyset$ is any $\mathbf{v} \in \mathsf{P}$. Given $\mathbf{v} \in \mathbb{R}^n$, we can easily

check whether $\mathbf{A}\mathbf{v} \leq \mathbf{b}$ and thus, $\mathbf{v} \in \mathsf{P}$. However, is there also a simple certificate for the case that $\mathsf{P} = \emptyset$? Indeed, there is such a certificate (see Schrijver, 1986, Corollary 7.1e):

Lemma 4 (Farkas' lemma) *The system $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ is solvable iff every $\boldsymbol{u} \in \mathbb{R}^d$ with $\boldsymbol{A}^\intercal \boldsymbol{u} = 0$, $\boldsymbol{u} \geq 0$ satisfies $\boldsymbol{u}^\intercal \boldsymbol{b} \geq 0$.*

Thus, any $\mathbf{u} \in \mathbb{R}^d$ satisfying $\mathbf{A}^\intercal \mathbf{u} = 0$, $\mathbf{u} \geq 0$ and $\mathbf{u}^\intercal \mathbf{b} < 0$ gives a certificate that P is empty. Indeed, given $\mathbf{x} \in \mathsf{P}$, write $0 = 0^\intercal \mathbf{x} = (\mathbf{A}^\intercal \mathbf{u})^\intercal \mathbf{x} = (\mathbf{u}^\intercal \mathbf{A})\mathbf{x} = \mathbf{u}^\intercal (\mathbf{A}\mathbf{x}) \leq \mathbf{u}^\intercal \mathbf{b} < 0$ to get a contradiction.

To decide using a computer whether $\mathsf{P} \neq \emptyset$ or not, we can make use of well-developed tools from the theory of linear programming (see below, Section 3.4).

3.2 Dimension and Faces of a Polyhedron

The *dimension* $\dim(\mathsf{P})$ of a polyhedron $\mathsf{P} \subseteq \mathbb{R}^n$ is the dimension of its *affine hull* $\mathrm{aff}(\mathsf{P})$, which is the set of all finite affine combinations $\sum_t \lambda_t \cdot \mathbf{v}_t$ (with $\lambda_t \in \mathbb{R}$ for all t and $\sum_t \lambda_t = 1$) of elements $\mathbf{v}_t \in \mathsf{P}$. By convention, the dimension of the empty set is -1. A polyhedron is *full-dimensional* if $\dim(\mathsf{P}) = n$.

Given $\mathbf{v} \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$, we call the inequality $\langle \mathbf{v}, \mathbf{x} \rangle \leq \alpha$ *valid* for a polyhedron $\mathsf{P} \subseteq \mathbb{R}^n$ if it is satisfied for every $\mathbf{x} \in \mathsf{P}$. If the inequality $\langle \mathbf{v}, \mathbf{x} \rangle \leq \alpha$ is valid for the polyhedron P, we call the set $\mathsf{F} = \mathsf{P} \cap \{\mathbf{x} \in \mathbb{R}^n; \langle \mathbf{v}, \mathbf{x} \rangle = \alpha\}$ a *face* of P.

The sets $\emptyset$ and P are always faces of a polyhedron P, defined by the valid inequalities $\langle 0, \mathbf{x} \rangle \leq 1$ and $\langle 0, \mathbf{x} \rangle \leq 0$, respectively. Every polyhedron P has only finitely many faces and all of them are polyhedra, too (see Schrijver, 1986, § 8.3). They can be classified by their dimension. Faces of dimension 0 are points in $\mathbb{R}^n$, called *vertices*. Faces of dimension 1 are called *edges*: these are either line-segments, half-lines or lines. Faces of dimension $\dim(\mathsf{P}) - 1$ are called *facets*.

If a polyhedron is full-dimensional then facet-defining inequalities establish a unique (up to positive scalar factors) inclusion-minimal inequality system defining P (for details see Schrijver, 1986, § 8.4). Since every (proper) face F of P is the intersection of facets containing it, facets implicitly determine all faces.

A polyhedron $\mathsf{P} = \{\mathbf{x}; \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ is *pointed* if the only vector $\mathbf{y}$ with $\mathbf{A}\mathbf{y} = 0$ is the zero vector $\mathbf{y} = 0$ (see Schrijver, 1986, § 8.2). A non-empty pointed polyhedron has at least one vertex. A pointed polyhedral cone C has just one vertex 0, its edges are half-lines, called *extreme rays*. Non-zero representatives of the extreme rays of C then provide its inclusion-minimal inner description (Schrijver, 1986, § 8.8).

3.3 Hilbert Basis

A *lattice point* (in $\mathbb{R}^n$) is an integral vector $\mathbf{x} \in \mathbb{Z}^n$, that is, a vector whose components are integers. If C is a pointed rational polyhedral cone then each of its extreme rays contains a non-zero lattice point; this lattice point is unique if it is *normalized*, that is, if its components have no common integer divisor. Then one can show that every lattice point in C is a linear combination of these unique representatives (of extreme rays) with non-negative rational coefficients (Studený, 1993, Lemma 10). However, in general, not every lattice point in C can be obtained as such a combination of those representatives with non-negative *integral* coefficients. Therefore, the following concept is important.

By a *Hilbert basis* of a rational polyhedral cone C (see Schrijver, 1986, § 16.4) is meant a finite set $\mathsf{H} \subseteq \mathsf{C}$ of lattice points such that every lattice point in C is a non-negative integer linear combination of elements in H. A relevant result in polyhedral geometry (Schrijver, 1986, Theorem 16.4) says that every rational polyhedral cone possesses a Hilbert basis. If the cone is in addition pointed, there is a unique inclusion-minimal Hilbert basis. Clearly, any Hilbert basis of C has to include the above mentioned normalized representatives of extreme rays of C.

Again, it is a challenging algorithmic task to compute the minimal integral Hilbert basis of a given (pointed) rational polyhedral cone. However, there are a few software packages that allow to do so in some simpler cases, for example 4TI2 (4ti2 team, 2008) or NORMALIZ (Bruns, Ichim, and Söger, 2009).

3.4 LP Problems and the Simplex Method

A *linear programming* (LP) problem is the task to maximize/minimize a linear function $\mathbf{x} \mapsto \langle \mathbf{c}, \mathbf{x} \rangle$, where $\mathbf{c} \in \mathbb{R}^n$, over a polyhedron $\mathsf{P} = \{\mathbf{x} \in \mathbb{R}^n;\, \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$. A classic tool to deal with this problem is the *simplex method*. Nevertheless, it is not the aim to describe here the details of this method; they can be found in textbooks on linear programming; for example Schrijver (1986, Chapter 11).

We just recall its main features to explain the geometric interpretation. The plain version of the simplex method (see Schrijver, 1986, § 11.1) assumes one has a vertex of (a pointed polyhedron) P as a starting iteration. The basic idea is to move from vertex to vertex along the edges of P until an optimal vertex is reached or an edge is found on which the linear function is unbounded. Each particular variant of the simplex method uses a special *pivoting operation* to choose the next iteration, which should have a better value of the linear objective function than the previous one.

However, it may be the case that one is not sure whether the polyhedron P is non-empty. Then the simplex method has two phases. In Phase I, one finds at least one $\mathbf{x} \in \mathsf{P}$, called a *feasible vertex solution*, provided that it exists, or one concludes that $\mathsf{P} = \emptyset$ otherwise. If a feasible solution is available, Phase II consists of finding an optimal solution as mentioned above. A standard trick to deal with Phase I is to re-formulate it as an auxiliary LP problem, in which a feasible vertex solution is known.

For example, consider the task to optimize a linear function over a special polyhedron of the form $\mathsf{Q} = \{\mathbf{x} \in \mathbb{R}^n;\, \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathsf{0}\}$, where $\mathbf{A} \in \mathbb{R}^{d \times n}$ and $\mathbf{b} \in \mathbb{R}^d$, which task is, in fact, equivalent to the general case (see Schrijver, 1986, § 7.4). Moreover, one can assume without loss of generality $\mathbf{b} \geq \mathsf{0}$ here.[2] Then the auxiliary LP problem is as follows:

$$\min \{\, \langle \mathsf{1}, \mathbf{z} \rangle;\ (\mathbf{x}, \mathbf{z}) \in \mathbb{R}^{n+d},\ \mathbf{A}\mathbf{x} + \mathbf{z} = \mathbf{b},\ \mathbf{x} \geq \mathsf{0},\ \mathbf{z} \geq \mathsf{0} \,\}. \tag{4}$$

This LP problem has the vector $(\mathsf{0}, \mathbf{b})$ as a feasible vertex solution[3] and it has the property that its optimal value $\langle \mathsf{1}, \mathbf{z} \rangle$ is 0 iff $(\mathbf{x}, \mathbf{z}) = (\mathbf{x}, \mathsf{0})$ and $\mathbf{x} \in \mathsf{Q}$. Thus, testing whether the polyhedron Q is non-empty can be done in this way.

4. Methods

In this section, we describe a few methods that can be used to test CI inference. The task is as follows. An input list L of *elementary* CI statements and another elementary CI statement t over

2. Indeed, if the i-th component of $\mathbf{b}$ is negative, one can replace the i-th row of $\mathbf{A}$ and the i-th component of $\mathbf{b}$ with their multiples by -1.

3. Actually, $(\mathsf{0}, \mathbf{b})$ is a vertex of the region, defined by valid inequality $\langle \mathsf{1}, \mathbf{x} \rangle \geq 0$.

N are given and the goal is to decide whether $u_L \rightharpoonup u_t$ (cf., Proposition 1). We are going to give a computational comparison of these methods for $|N| = 5$ in Section 5.1 to see which of them is the fastest.

4.1 Method 1: using Skeletal Characterization

This is the first ever implemented method for testing independence implication (Studený et al., 2000, § 7.4). The motivation source for this method was the dual definition (2) of independence implication in terms of supermodular functions (see Section 2.3). The point is that, in (2), written in contrapositive form as [4]

$$\langle m, v\rangle > 0 \quad \Longrightarrow \quad \langle m, u\rangle > 0\,,$$

one can limit oneself to ℓ-standardized supermodular functions $m : \mathcal{P}(N) \to \mathbb{R}$, that is, to those that satisfy $m(S) = 0$ if $|S| \leq 1$. This is a pointed rational polyhedral cone. The class of normalized integral representatives of its extreme rays was called the *ℓ-skeleton* in Studený (2005) and denoted by $\mathcal{K}^{\diamond}_{\ell}(N)$.

The *skeletal characterization* of the implication $u \rightharpoonup v$, for $u, v \in \mathcal{S}(N)$, is as follows:

$$\forall\, m \in \mathcal{K}^{\diamond}_{\ell}(N) \qquad \langle m, v\rangle > 0 \quad \Longrightarrow \quad \langle m, u\rangle > 0 \tag{5}$$

(see Studený, 2005, Lemma 6.2). The ℓ-skeleton was computed for $|N| = 5$ (Studený et al., 2000): it consists of 117978 imsets, which break into 1319 permutational types.

Thus, the corresponding algorithm consists in checking whether each of the implications in (5) is valid or not. To disprove the implication $u \rightharpoonup v$ it is enough to find at least one $m \in \mathcal{K}^{\diamond}_{\ell}(N)$ violating (5); to confirm it one has to check that all of them are valid. By ordering the elements of the ℓ-skeleton such that those $m \in \mathcal{K}^{\diamond}_{\ell}(N)$ which are more likely to cause violation are tried earlier one can speed up the disproval (but not the confirmation). This suggested to sort elements of $\mathcal{K}^{\diamond}_{\ell}(N)$ on the basis of zeros in $\{\langle m, w\rangle\,;\; w \in \mathcal{E}(N)\}$ (for details see Bouckaert and Studený, 2007, § 3.1).

Example 1 *To illustrate the method consider a trivial example with $N = \{a, b, c\}$: take the input $L = \{\, a \perp\!\!\!\perp b \,|\, c\,,\; a \perp\!\!\!\perp c \,|\, \emptyset\,\}$ and $t : a \perp\!\!\!\perp b \,|\, \emptyset$. We already know by semi-graphoid properties that L probabilistically implies t (see Section 2.1). However, the aim of this example is to get this conclusion through Proposition 1 (see Section 2.3) using the skeletal characterization. We have*

$$u_L = u_{\langle a,b|c\rangle} + u_{\langle a,c|\emptyset\rangle} = \delta_{abc} - \delta_{bc} - \delta_a + \delta_\emptyset \quad \textit{and} \quad u_t = u_{\langle a,b|\emptyset\rangle} = \delta_{ab} - \delta_a - \delta_b + \delta_\emptyset\,.$$

In the case of 3 variables, the ℓ-skeleton has 5 elements, listed in rows of Table 1. The columns in the table correspond to structural imsets u_t and u_L, the items are corresponding scalar products. The condition (5) evidently holds for $v = u_t$ and $u = u_L$. Thus, $u_L \rightharpoonup u_t$.

The interpretation of the method from the point of view of polyhedral geometry is as follows. The independence implication can equivalently be defined in terms of (inclusion of) facets of the cone $\mathcal{R}(N) \equiv \mathrm{cone}\,(\mathcal{E}(N))$, the cone spanned by elementary imsets. Let F_u denote the face of $\mathcal{R}(N)$ generated by $u \in \mathcal{S}(N) \subseteq \mathcal{R}(N)$, that is, the least face of $\mathcal{R}(N)$ containing u ($\equiv$ the intersection of all faces of $\mathcal{R}(N)$ containing u). Then, for $u, v \in \mathcal{S}(N)$, one has $u \rightharpoonup v$ iff the face F_u contains v, which means, $F_v \subseteq F_u$ (see Studený, 2005, Remark 6.2).

4. Note that $\langle m, u\rangle \geq 0$ for any m supermodular and $u \in \mathcal{S}(N)$ (see Studený, 2005, Proposition 5.1).

	$u_t = \delta_{ab} - \delta_a - \delta_b + \delta_\emptyset$	$u_L = \delta_{abc} - \delta_{bc} - \delta_a + \delta_\emptyset$
$m_1 = \delta_{abc}$	0	1
$m_2 = \delta_{abc} + \delta_{ab}$	1	1
$m_3 = \delta_{abc} + \delta_{ac}$	0	1
$m_4 = \delta_{abc} + \delta_{bc}$	0	0
$m_5 = 2 \cdot \delta_{abc} + \delta_{ab} + \delta_{ac} + \delta_{bc}$	1	1

Table 1: Scalar products with elements of $\mathcal{K}^\diamond_\ell(N)$ for $N = \{a,b,c\}$.

The method is based on the computation of all facets of the cone $\mathcal{R}(N)$. These facets correspond to the extreme rays of the (dual) cone of ℓ-standardized supermodular functions. Thus, basically, one is checking whether every facet containing F_u also contains F_v. The problem with this approach is that it can hardly be extended beyond five variables because computing these facets seems to be computationally infeasible for $|N| \geq 6$.

4.2 Method 2: Racing Algorithms

The idea of the paper (Bouckaert and Studený, 2007) was to combine two algorithms for testing the independence implication $u \rightharpoonup v$. One of them, called the *verification algorithm*, was based on (1) and appeared to be suitable to confirm the implication provided it holds. However, it may spend a long time before it gives a response if the implication does not hold. The other algorithm, called the *falsification algorithm* and based on (2), was designed to disprove the implication if it does not hold. However, it is not able to confirm $u \rightharpoonup v$ provided it holds.

The combined procedure starts with two threads, the verification one and the falsification one. Once one thread finds its proof, it stops the other and returns its outcome. This approach makes it possible to go beyond 5 variables, but may not give a decisive response (in reasonable time) to some complex implication problems. On the other hand, empirical evidence from Bouckaert and Studený (2007) suggests that this method is, on average, faster than the method described in Section 4.1.

4.2.1 VERIFICATION: DECOMPOSING INTO ELEMENTARY IMSETS

Consider a combinatorial imset $u \in \mathcal{C}(N)$, an elementary imset $v \in \mathcal{E}(N)$ and the task to decide whether $u \rightharpoonup v$. That is, by (1), testing whether $k \cdot u - v$ is a structural imset for some $k \in \mathbb{N}$. Observe that (1) is equivalent to

$$\exists l \in \mathbb{N} \qquad l \cdot u - v \in \mathcal{C}(N)\,. \tag{6}$$

Indeed, $\mathcal{C}(N) \subseteq \mathcal{S}(N)$ gives (6)$\Rightarrow$(1). Now, (1) implies that $n \cdot (k \cdot u - v)$ is a combinatorial imset for some $k, n \in \mathbb{N}$ (see Section 2.2). As $(n-1) \cdot v \in \mathcal{C}(N)$, it gives $(n \cdot k) \cdot u - v \in \mathcal{C}(N)$. Moreover, it follows from concluding remarks in Sections 2.2 and 2.3 that $n_* \cdot k_*$ is an upper limit for l in (6). In general, we do not know what is the least such upper limit for l. Even in case $|N| = 5$, we only know it is a number between 7 (see Studený et al., 2000) and $14 = 2 \cdot 7 = n_* \cdot k_*$.

The characterization (6) allows one to transform testing independence implication to the task to decide whether a given candidate imset $y = l \cdot u - v$ is combinatorial. A combinatorial imset y may have many decompositions $y = \sum_{w \in \mathcal{E}(N)} k_w \cdot w$, $k_w \in \mathbb{Z}_+$ into elementary imsets. However, the number $\sum_{w \in \mathcal{E}(N)} k_w$ of summands, called the *degree* of y, is the same for any such decomposition (see Studený, 2005, § 4.2.2). Because there is a simple formula for the degree of the candidate imset

y, the search space, the tree of potential decompositions, is known. This space could be big, but it can be limited by introducing additional sanity checks, which allow one to cut off some blind alleys in the tree. Moreover, the search can be guided by suitable heuristics and this can speed up the resulting algorithm (for details see Bouckaert and Studený, 2007, § 3.2).

The decomposition itself is quite fast, but what can slow down the whole procedure is the factor l from (6), depending on a particular pair u, v. The point is that the degree of a candidate imset $y = l \cdot u - v$ grows (linearly) with l; consequently, the size of the corresponding tree of possible decompositions grows exponentially with l. Typically, if $u \rightharpoonup v$ holds then one often finds a decomposition of y with $l = 1$. However, for $|N| = 5$, there are a few cases when the decomposition of y exists for $1 < l \leq 7$ and not for $l - 1$ (for examples see Studený et al., 2000, § 4.3). In these cases and also when $u \rightharpoonup v$ does not hold one has to search through a huge space, which makes the method infeasible for efficient disproving implications.

Example 2 *Consider $N = \{a, b, c, d\}$, the input list*

$$L = \{\, a \perp\!\!\!\perp b \,|\, c\,,\ a \perp\!\!\!\perp c \,|\, d\,,\ a \perp\!\!\!\perp d \,|\, b\,,\ b \perp\!\!\!\perp c \,|\, ad \,\}\,, \tag{7}$$

and another CI statement $t : a \perp\!\!\!\perp c \,|\, b$. We are going to show $u_L \rightharpoonup u_t$ by the decomposition method. Actually, we show that (6) holds with $l = 1$. More specifically, we have

$$\begin{aligned} u_L &= u_{\langle a,b|c\rangle} + u_{\langle a,c|d\rangle} + u_{\langle a,d|b\rangle} + u_{\langle b,c|ad\rangle} \\ &= \delta_{abcd} + \delta_{abc} - \delta_{ab} - \delta_{ac} - \delta_{bc} - \delta_{bd} - \delta_{cd} + \delta_b + \delta_c + \delta_d\,, \end{aligned}$$

and, as $u_t = \delta_{abc} - \delta_{ab} - \delta_{bc} + \delta_b$, we know that

$$y \equiv 1 \cdot u_L - u_t = \delta_{abcd} - \delta_{ac} - \delta_{bd} - \delta_{cd} + \delta_c + \delta_d\,.$$

The task is to test whether y is a combinatorial imset. If this is the case then the degree of y must be 3, which means we search for a decomposition into elementary imsets with 3 summands. Clearly, at least one summand v has to satisfy $v(abcd) > 0$. There are 6 elementary imsets over $\{a, b, c, d\}$ with this property and two of them are excluded by sanity checks. For example, for $v' = u_{\langle c,d|ab\rangle}$ and $y' = y - v'$ one has $-1 = \sum_{\{c,d\} \subseteq T} y'(T) < 0$, which is impossible for a combinatorial imset in place of y'. However, if we subtract $\tilde{v} = u_{\langle a,b|cd\rangle}$ then

$$\tilde{y} = y - \tilde{v} = \delta_{acd} + \delta_{bcd} - \delta_{ac} - \delta_{bd} - 2 \cdot \delta_{cd} + \delta_c + \delta_d$$

is a good direction. Again, at least one of two summands v in the searched decomposition of $\tilde{y}$ must satisfy $v(acd) > 0$ and the choice $v = u_{\langle a,d|c\rangle}$ leads to the final decomposition

$$y = u_{\langle a,b|cd\rangle} + u_{\langle a,d|c\rangle} + u_{\langle b,c|d\rangle}\,.$$

Thus, the implication $u_L \rightharpoonup u_t$ has been confirmed by the decomposition method.

To explain the geometric interpretation (of the method) note that the cone $\mathcal{R}(N) = \mathrm{cone}\,(\mathcal{E}(N))$ is slightly special. The lattice points in this cone are just structural imsets.[5] Moreover, there exists

5. As mentioned in Section 3.3, lattice points in $\mathcal{R}(N)$ are just non-negative rational combinations of elementary imsets $\mathcal{E}(N)$, that is, structural imsets (see Section 2.2).

a hyperplane in $\mathbb{R}^{\mathcal{P}(N)}$ which intersects all extreme rays of $\mathcal{R}(N)$ just in its normalized integral representatives $\mathcal{E}(N)$ and these are the only lattice points in the intersection of this hyperplane with $\mathcal{R}(N)$. In particular, every structural imset belongs to one of parallel hyperplanes (to this basic one) and its "degree" says how far it is from the origin (= zero imset). Now, the condition (1) means that at least one multiple of u (by $k \in \mathbb{N}$) has the property that it remains a lattice point within the cone $\mathcal{R}(N)$ even if v is subtracted. The condition (6) is a minor modification of (1): it requires a multiple of u minus v is a sum of elementary imsets (with possible repetition). The algorithm, therefore, looks for the decomposition and the degree (of the candidate imset y) serves as the measure of its complexity.

4.2.2 FALSIFICATION: RANDOMLY GENERATING SUPERMODULAR FUNCTIONS

Falsification is based on the characterization (2). To disprove the implication $u \rightharpoonup v$ it is enough to find a supermodular function $m : \mathcal{P}(N) \to \mathbb{R}$ such that $\langle m, u \rangle = 0$ and $\langle m, v \rangle > 0$. Actually, one can limit oneself to ℓ-standardized integer-valued supermodular functions, that is, *supermodular imsets*. These imsets have a special form which allows one to generate them randomly. The idea is

- to randomly generate a collection of subsets of N,
- to assign them randomly selected positive integer values (and the zero values to remaining sets), and
- to modify the resulting function to make it a supermodular imset.

The details of this procedure can be found in Bouckaert and Studený (2007, § 3.3). The procedure allows one to disprove (= falsify) the implication $u \rightharpoonup v$ even for $|N| \geq 6$, but it is clearly not able to confirm it provided it holds. Therefore, it has to be combined with a verification procedure.

Example 3 *Consider $N = \{a,b,c,d\}$ and the same input list (7) as in Example 2, but a different CI statement $t : b \perp\!\!\!\perp c \,|\, a$ to be derived. If our random procedure generates the supermodular imset*

$$m = 2 \cdot \delta_{abcd} + \delta_{abc} + \delta_{abd} + \delta_{acd} + 2 \cdot \delta_{bcd} + \delta_{bc} + \delta_{bd} + \delta_{cd} \,,$$

then one can observe that $\langle m, u_L \rangle = 0$ while $\langle m, u_{\langle b,c|a \rangle} \rangle = 1$. In particular, by (2), the respective implication is not valid: $\neg(u_L \rightharpoonup u_{\langle b,c|a \rangle})$.

The geometric interpretation of the algorithm is similar to the interpretation of the method from Section 4.1. Supermodular functions correspond to faces of the cone $\mathcal{R}(N)$. Thus, the procedure consists in random generating faces of $\mathcal{R}(N)$ and the aim to find a face of $\mathcal{R}(N)$ which contains u but not v.

4.3 Method 3: Decomposition via Hilbert Basis

An alternative to testing whether an imset is combinatorial is testing whether it is structural. Since structural imsets coincide with the lattice points in the cone $\mathcal{R}(N) \equiv \text{cone}\,(\mathcal{E}(N))$, each of them can be written as a sum (with possible repetition) of the elements of the Hilbert basis $\mathcal{H}(N)$ of the cone $\mathcal{R}(N)$ (see Section 3.3). For $|N| \leq 4$ one has $\mathcal{H}(N) = \mathcal{E}(N)$ (Studený, 1991); however, the results from Hemmecke et al. (2008) imply that the set of elementary imsets does not constitute a Hilbert basis of $\mathcal{R}(N)$ for $|N| \geq 5$.

Computation of the Hilbert basis is a very hard task. Recently, the Hilbert basis of $\mathcal{R}(N)$ for $|N| = 5$ has been obtained as a result of sophisticated computations (Bruns et al., 2010). In Appendix B we provide a list of its representatives. Altogether, the obtained Hilbert basis of $\mathcal{R}(N)$ has 1300 elements, falling into 35 (permutation equivalence) types.

Thus, having the Hilbert basis of $\mathcal{R}(N)$ at hand, one can test the independence implication $u \rightharpoonup v$ for $u, v \in \mathcal{S}(N)$ through (1): the task is to find out whether there exists a decomposition of $y \equiv k \cdot u - v$ into Hilbert basis elements for some $k \in \mathbb{N}$. This is analogous to the decomposition approach from Section 4.2.1, where the set of elementary imsets $\mathcal{E}(N)$ was used instead of $\mathcal{H}(N)$.

Thus, the interpretation of this new method is the same, the difference is that we have now a wider class of leaves of the (potential) decomposition tree. On the other hand, the advantage of the Hilbert basis decomposition for $|N| = 5$ should be that, in some complex cases, a much simpler decomposition of y may exist that involves also the elements of $\mathcal{H}(N) \setminus \mathcal{E}(N)$ (simpler = with a less number of summands). We are also sure that the upper limit for the constant k is only $k_* = 7$. In particular, the depth of the tree of potential decompositions should be smaller, while the tree itself is expected to be wider.

4.4 Method 4: Linear Programming

The basic idea is to re-formulate (the definition of) independence implication in terms of the (pointed rational polyhedral) cone $\mathcal{R}(N) \equiv \mathrm{cone}(\mathcal{E}(N))$ spanned by elementary imsets. More specifically, given $u, v \in \mathcal{S}(N)$, the condition (1) can be expressed in this way:

$$u \rightharpoonup v \quad \text{iff} \quad \exists k \in [0, \infty) \quad k \cdot u - v \in \mathcal{R}(N)\,. \tag{8}$$

Indeed, since $\mathcal{S}(N) \subseteq \mathcal{R}(N)$ the implication (1)⇒(8) is evident. Conversely, provided (8) holds with k, it holds with any $k' \geq k$ because $\mathcal{R}(N)$ is a cone and $(k' - k) \cdot u \in \mathcal{R}(N)$. Therefore, there exists $k' \in \mathbb{N}$ with $k' \cdot u - v \in \mathcal{R}(N)$. As $k' \cdot u - v$ is an imset, it belongs to $\mathcal{S}(N)$, and (1) holds.

The geometric interpretation of the condition (8) is clear. It means that the ray (= half-line) with the origin in $-v$ and the direction given by u intersects the cone $\mathcal{R}(N)$. The point is that testing whether this happens can be viewed as an LP problem:

Lemma 5 Given $u, v \in \mathcal{S}(N)$ one has $u \rightharpoonup v$ iff the system of equalities

$$\sum_{w \in \mathcal{E}(N)} \lambda_w \cdot w(S) - k \cdot u(S) = -v(S) \qquad \text{for any } S \subseteq N\,, \tag{9}$$

has a non-negative solution in λ_w, $w \in \mathcal{E}(N)$ and k.

Proof The cone $\mathcal{R}(N)$ consists of conic combinations of representatives of its extreme rays, that is, of elementary imsets. Thus, (8) can be re-written as the requirement for the existence of $k \geq 0$ and $\lambda_w \geq 0$ with $k \cdot u - v = \sum_{w \in \mathcal{E}(N)} \lambda_w \cdot w$. This imset equality, specified for any $S \subseteq N$, yields (9). ■

Non-negative solutions to (9) form a polyhedron of the type $\mathsf{Q} = \{\mathbf{x}; \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathsf{0}\}$, $\mathbf{A} \in \mathbb{R}^{d \times n}$, $\mathbf{b} \in \mathbb{R}^d$. Indeed, the rows of $\mathbf{A}$ correspond to subsets of N, while the columns to elementary imsets and the factor k. Thus, $d = |\mathcal{P}(N)| = 2^{|N|}$ and $n = |\mathcal{E}(N)| + 1 = \binom{|N|}{2} \cdot 2^{|N|-2} + 1$. As explained in Section 3.4, testing whether Q is non-empty is equivalent to solving the auxiliary LP problem (4) in

$(n+d)$-dimensional space. An illustrative example of the application of this LP method in the case $|N| = 3$ is given in Appendix C.

The advantage of this approach is that it is not limited to a particular number of variables as the previous ones. To get an impression of the implication problem complexity at hand, let us have a look at Table 2. Clearly, these (comparably small) linear programs should be solvable quickly in practice. We give computational evidence to this claim in Section 5.2. Another comment is that this LP method can be interpreted as a kind of decomposition procedure (analogous to the one from Section 4.2.1); see Appendix D for an explanation.

$\|N\|$	3	4	5	6	7	8	9	10
$n+d$	15	41	113	305	801	2049	5121	12545

Table 2: The dimensions of LP problems to be solved in order to decide $u \rightharpoonup v$.

Let us finally note that there are alternative ways to re-formulate testing of $u \rightharpoonup v$ as an LP problem. For example, consider the cone of ℓ-standardized supermodular functions. We know the outer description of this (pointed) cone:

$$\mathcal{K}_\ell(N) = \{m \in \mathbb{R}^{\mathcal{P}(N)};\ m(S) = 0 \text{ for } |S| \leq 1,\ \langle m, w\rangle \geq 0 \text{ for } w \in \mathcal{E}(N)\}.$$

Since, in (2), one can limit oneself to ℓ-standardized supermodular functions, $u \rightharpoonup v$ is equivalent to the requirement

$$\sup\{\langle m, v\rangle;\ m \in \mathcal{K}_\ell(N),\ \langle m, u\rangle = 0\} = 0. \tag{10}$$

This is an LP problem with a feasible region.[6] Thus, Phase II of the simplex method can be applied to solve it. Note that (10), mentioned in Studený (2004, § 5), can be viewed, after appropriate modifications, as the dual LP problem to (9) in the sense of the LP theory; however, we omit the details in this paper.

5. Experiments

In this section, we describe the results of our computational experiments. We performed two separate bunches of experiments. One of them was done in New Zealand and the aim was to compare various methods in case $|N| = 5$ (see Section 5.1).

As the result was that the LP method performs best in case $|N| = 5$, the aim of the other group of experiments (see Section 5.2), done in Germany, was to test the LP approach in case $|N| > 5$. These latter experiments were based on the commercial optimization software CPLEX (IBM Ilog team, 2009).

5.1 Comparison for Five Variables

Empirical evaluation of the methods can give insight in the practical behavior of the various methods. First, we considered the case of five variables, so that we can compare the new methods with techniques based on skeletal representations (see Section 4.1). The experimental set up from Bouckaert and Studený (2007) was used for the five-variable tests. In short, a thousand random input lists

6. Indeed, $m \equiv 0$ is a feasible solution to (10).

containing 3, up to 11 elementary CI statements were generated and, for each elementary CI statement outside the input list, it was verified whether it was implicated or not. So, the thousand 3-input cases result in verification of a thousand times the total number of elementary CI statements (80 for 5 variables) minus the 3 statements already given in the input list, that is, $1000 \times (80-3) = 77000$ inference problems. Table 3 lists the numbers of inference problems in its last column.

The algorithms considered were:

- Skeletal algorithm and sorted skeletal algorithm (see Section 4.1).
- Racing algorithms (see Section 4.2). There are situations where both the verification and the falsification algorithm perform poor, resulting in unacceptable running times. These outliers distort the typical behavior of the algorithm, so we put a cap on total calculation time and report the number of problems where a solution is not found within this deadline as the number of unresolved cases in Table 3.
- *Hilbert basis* (HB) decomposition algorithm (see Section 4.3). Initial experimentation showed that the HB approach is very good at verifying the implication. However, when a CI statement is not implied, it takes a long time to traverse the search space. So, the average time for the HB approach is expected to be very poor. Again, we capped the allowed time for the algorithm to run and when no solution was found within that time the problem was as marked unresolved.
- The observation that the HB algorithm performs well for implication but poor on falsification immediately gives rise to another algorithm where the HB decomposition algorithm and falsification algorithm are raced against each other. This algorithm is called the *Hilbert racer*. Like the racer algorithm, this algorithm was time constrained.
- Now we have two algorithms that appear to perform well for verification, it is natural to combine them with the falsifier and thus get a three horse race. This algorithm is called the *triple racer*, and it is time constrained as well.
- The last algorithm under consideration is the LP method described in Section 4.4.

Input size	Hilbert capped		Hilbert racer		racer		triple racer		Total
3	5127	(6.7%)	0	(0%)	0	(0%)	0	(0%)	77000
4	10098	(13.3%)	0	(0%)	0	(0%)	0	(0%)	76000
5	12793	(17.1%)	0	(0%)	6	(0.01%)	9	(0.01%)	75000
6	15139	(20.5%)	11	(0.01%)	28	(0.04%)	30	(0.04%)	74000
7	16475	(22.6%)	38	(0.05%)	110	(0.15%)	88	(0.12%)	73000
8	18042	(25.1%)	85	(0.12%)	238	(0.33%)	215	(0.30%)	72000
9	18308	(25.8%)	181	(0.25%)	377	(0.53%)	306	(0.43%)	71000
10	17738	(25.3%)	211	(0.30%)	489	(0.70%)	386	(0.55%)	70000
11	16798	(24.3%)	230	(0.33%)	495	(0.72%)	349	(0.51%)	69000

Table 3: Numbers of inference problems that were not resolved due to time-outs.

All algorithms were implemented in Java, run under Sun Java 1.6.0_12 in server mode on a Intel dual Core 3.00GHz CPU with 2GB memory, though memory is not an issue for any of the algorithms.

5.1.1 RESULTS AND DISCUSSION

Table 3 shows the number of unresolved cases when time was capped at 1 second per problem. Only the algorithms that did not resolve all problems are shown there. The numbers in brackets are percentages, the last column the total number of problems for a particular input size.

Given the total number of problems (around 70 thousand per input size) this restricted calculation took around 20 hours per input size. Clearly, the HB decomposition method when capped leaves a large proportion of implication problems unresolved, up to a quarter for the larger input sizes.

Combining the HB algorithm with the falsification algorithm reduces the number of unresolved problems to less than a third of a percent. Comparing the Hilbert racer with the original racer algorithm shows that more problems are resolved in the time available, so the HB decomposition algorithm appears to perform better at resolving problems. Finally, the triple racer, which combines both algorithms, performs in between the two algorithms in terms of number of unresolved problems. At first sight, one would expect the triple racer to perform at least as good as the Hilbert racer, however, since the test was run on a dual core processor instead of a machine with at least three cores, the various algorithms were too busy battling for processor access and lost time to make the deadline.

	User time						
Input size	skeletal	skeletal sorted	Hilbert	racer	Hilbert racer	triple racer	LP
3	286	351	5955	132	14	163	29
4	680	648	10320	207	23	318	31
5	1476	1122	12868	414	107	667	32
6	2748	1713	15225	707	1418	1050	32
7	5385	2652	16584	1690	5601	1915	32
8	8583	3625	18171	2884	12145	3159	32
9	12086	4771	18476	6099	24274	4602	32
10	16180	6439	18089	19238	28227	11919	31
11	19530	8225	17507	20962	32452	17145	31
	Elapsed time						
Input size	skeletal	skeletal sorted	Hilbert	racer	Hilbert racer	triple racer	LP
3	288	353	12745	90	28	152	31
4	681	650	16471	162	38	277	32
5	1477	1124	18636	313	105	562	33
6	2749	1714	20606	516	876	855	33
7	5385	2652	21665	1016	3120	1548	34
8	8582	3625	22977	1889	6864	2522	34
9	12085	4771	23156	4610	13509	3693	33
10	16177	6439	22749	17080	16350	10509	33
11	19528	8224	22118	18255	18221	15250	32

Table 4: Total computer times for the experiments in seconds.

Table 4 shows total computer time to resolve all problems for a given input size. This is a rough indication of the average computation time per implication problem, given that the number of problems per input size changes slightly (less than 10%) from the smallest 3-input problems to the largest 11-input problems. The user time is time the processor spent on the problem, while elapsed

time is actual clock time. Since the racing algorithms are multi-threaded, having multiple jobs run in parallel, elapsed time can be less than user time. For the single threaded algorithms, user time is approximately equal to elapsed time.

Comparing the unsorted and sorted skeleton approaches, it shows that ordering the skeletons can result in considerable time savings.

The HB decomposition algorithm is outperformed, clearly due to its bad performance in falsifications. There is a relatively large gap between user and elapsed time for the HB algorithm, which is due to the way the time-out is implemented. This causes the central processing unit to be idle for some of the time in some cases. Note the high correlation between user time and number of unresolved problems in Table 3. This indicates that the HB decomposition algorithm finds a resolution for a large number of problems very quickly, but performs very poor on a large number of others.

The racer algorithm performs very well on the smaller input sizes, but looses out on the larger ones. The Hilbert racer performs even better on the smaller problems, but again looses out on the larger input sizes. The triple racer is the best performer on the larger problems. However, taking in account the number of unresolved problems, the Hilbert racer is winning out, so it is to be expected that a slightly longer time-out period will bring the performance of the triple racer on the same level as the Hilbert racer.

The most remarkable result is the performance of the LP algorithm. Unlike all the other algorithms, it does not suffer from performance degradation with increasing input sizes. Furthermore, the calculation times are just a fraction of the times for the other algorithms. Considering that this is a straightforward Java implementation of the algorithm where only limited effort went in optimizing the code, its performance compared to the other algorithms is overwhelmingly better.

5.2 Experiments for Higher Number of Variables

To perform the experiments for $|N| > 5$, the only chance is to use the LP method presented in Section 4.4. The experiments were analogously defined as in Section 5.1. If L is an input list of elementary CI statements, then one single experiment comprises the testing of $|\mathcal{E}(N)|$ many independence implications $u_L \rightharpoonup u$ for all $u \in \mathcal{E}(N)$.[7] For $|N| = 4,5,6$ we have considered input lists L containing 3 up to 11 different elementary CI statements over N and performed 1000 such experiments for each combination of $|N|$ and $|L|$. Thus, we made 9000 experiments in total for $|N| = 4,5,6$. For $|N| = 7,8,9,10$ we only considered 1000 experiments with L containing 3 elementary CI statements.

The number of LP problems to be tested within a single experiment and the dimension of these problems depends on the number $|\mathcal{E}(N)| = \binom{|N|}{2} \cdot 2^{|N|-2}$ of elementary imsets and on the number $2^{|N|}$ of subsets of N (compare with Table 2 in Section 4.4). The running times are averaged over all 9000 experiments for $|N| = 4,5,6$ and over all 1000 experiments for $|N| = 7,8,9,10$, respectively. The running times include the set up of the LP method and the actual LP computation. The tests are done using the commercial optimization software CPLEX 9.100 (IBM Ilog team, 2009) on a Sun Fire V890 Ultra Sparc IV processor.

Table 5 illustrates the growth of the running times of the computations for $|N| = 4$ up to $|N| = 10$. Therein, we relate the running times with the growth of $|N|$ and the amount $|\mathcal{E}(N)|$ of LP problems to be tested for each experiment.

7. Contrary to Section 5.1, we involved "unnecessary" testing $u_L \rightharpoonup u_t$ for $t \in L$ in the experiments.

$\|N\|$	4	5	6	7	8	9	10
LPs/experiment	24	80	240	672	1792	4608	11520
time/experiment	4 ms	18 ms	115 ms	658 ms	5037 ms	150.38s	3862.10s

Table 5: The growth of computation times for $|N| = 4$ to $|N| = 10$.

5.2.1 RESULTS AND DISCUSSION

Table 5 illustrates a natural increase in the time needed for one experiment as $|N|$ increases. This is clear as both more LP problems have to be solved and they are of higher dimensions. For small values of $|N|$ the increase in time occurs mainly because of the LP set up, not because of the computation itself. For higher $|N|$ this proportion changes. But in spite of that, even for $|N| = 7$ all 1000 experiments together could be done within minutes. For $|N| = 10$ it took more than one month to perform all 1000 experiments; however, it took only about an hour for each single experiment. More specifically, on average, only about one third of a second was enough to test $u_L \rightharpoonup u_t$ for $|N| = 10$.

Just looking at the increase of the dimensions of the LP problems, it is reasonable to assume that one can test independence implication for up to $|N| = 15$ by directly plugging the corresponding LP problem into CPLEX. However, testing all $|\mathcal{E}(N)|$ inference problems for one experiment would be too expensive. Recall that one experiment already required about one hour of computation time for $|N| = 10$. To go even beyond $|N| = 15$ for testing $u_L \rightharpoonup u_t$, one can still exploit the known structure of the matrix $\mathbf{A}$ of the LP problem and employ column generation techniques in order to solve these much bigger LP problems to optimality. Of course, this approach has to be coded and tested for bigger $|N|$ to determine its efficiency.

6. Conclusions

Our computational experiments confirmed that the LP approach both overcomes the barriers of previous methods set by the number of variables in consideration and, moreover, it is also much faster. The new method also has the perspective to go even beyond the present limits, provided special LP techniques are used.

For small CI inference problems, involving a limited number of variables, one can enter data manually through web interfaces (based on LP method) available at

`http://www.cs.waikato.ac.nz/~remco/ci/.`

For bigger inference problems, in which the input is expected in the form of a file, one can use efficient commercial LP software instead.

We showed that the potential application of computer testing of CI inference is in the area of Bayesian networks (see Section 2.4). More specifically, by Proposition 2 (and Corollary 3), testing independence inclusion (and reading CI restrictions from an acyclic directed graph) can be transformed to the task whether a difference of two standard imsets is a combinatorial imset. Since, by (3), every standard imset has at most $2 \cdot |N|$ non-zero values, and, also, its degree is at most $|N| - 1$, we have good reasons to believe that the decomposition algorithm from Section 4.2.1 can be implemented with polynomial complexity in this special case. Actually, a relevant result (Studený and Vomlel, 2011, Lemma 3) implies that if the difference of two standard imsets is a combinatorial imset then it is a plain sum of elementary imsets (= a combination with coefficients $+1$). Thus, we

dare to conjecture that testing whether a difference of two standard imsets is a combinatorial imset can be done with polynomial complexity in $|N|$ using the procedure from Section 4.2.1.

After finishing this paper, we learned about a conference paper by Niepert (2009), which also comes with the idea of application of the LP approach to probabilistic CI inference. The reader may be interested in what is the relation of both approaches. The LP problem in Niepert (2009, Proposition 4.9) is, in fact, equivalent to our task (9) with fixed factor $k = 1$, if one uses a suitable one-to-one linear transformation to involved imsets u, v and w. This transformation has a nice property that the transformed LP problem $\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{b}}, \mathbf{x} \geq \mathbf{0}$ has 0-1 matrix $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{b}} \geq \mathbf{0}$. However, since the transformation is linear and one-to-one, it is essentially the same LP problem. Our approach is more general because it allows one to consider the multiplicative factor $k \geq 0$ in (9), which results in a wider class of derivable CI implications (cf., Section 4.2.1). We hope that the algorithms presented in our paper will find their application, perhaps in the research on stable CI statements (Niepert, van Gucht, and Gyssens, 2010).

The results presented in the paper lead to further open theoretical problems. The most difficult one is probably to find out what is the exact value of the constant n_* from Section 2.2 for $|N| > 5$, as it seems to be related to finding the Hilbert basis $\mathcal{H}(N)$, which is a hard task. Another group of open questions concerns the verification of the implication $u \rightharpoonup v$ for $u \in \mathcal{C}(N)$ and $v \in \mathcal{E}(N)$. We would like to know the minimal bounds for the factors in (1), (6) and (8). Perhaps computing these bounds can be formulated as a mixed (integer) LP problem. Finally, there are general questions of complexity, like whether it is NP hard to decide CI implications with respect to the input size.

Acknowledgments

The research of Milan Studený has been supported by the grants GAČR n. 201/08/0539 and MŠMT n. 1M0572. We thank the reviewers for their comments, which helped to improve the readability of the paper.

Appendix A. Proof of Proposition 1

Recall that L is an input list of CI statements over N and t another CI statement over N. The basic tool for our arguments is the *multiinformation function* $m_P : \mathcal{P}(N) \to \mathbb{R}$ ascribed to any discrete probability distribution P over N (for technical details see Studený, 2005, § 2.3.4). Corollary 2.2 in Studený (2005) says that m_P is always a supermodular function and, for every CI statement $s \equiv A \perp\!\!\!\perp B\,|\,C$ over N, one has

$$\langle m_P, u_s \rangle = 0 \quad \text{iff} \quad A \perp\!\!\!\perp B\,|\,C\,[P]\,. \tag{11}$$

Now, assuming $u_L \rightharpoonup u_t$, the equivalent characterization (2) of the independence implication implies that, for every discrete probability distribution P over N,

$$\text{whenever } \langle m_P, u_L \rangle = 0 \quad \text{then} \quad \langle m_P, u_t \rangle = 0\,. \tag{12}$$

Assuming P is a distribution such that every s in L is valid with respect to P, one has $\langle m_P, u_s \rangle = 0$ by (11), and, hence, by the linearity of the scalar product, $\langle m_P, u_L \rangle = \sum_{s \in L} \langle m_P, u_s \rangle = 0$. This implies, by (12), $\langle m_P, u_t \rangle = 0$, which means, by (11), that t is a valid CI statement with respect to P.

Appendix B. Hilbert Basis for Five Variables

As said in Section 4.3, the Hilbert basis $\mathcal{H}(N)$ for $|N| = 5$ falls into 35 types. Nevertheless, there is a further symmetry within it. Consider a bijective *reflection* map $\iota : \mathcal{P}(N) \to \mathcal{P}(N)$, given by $\iota(A) = N \setminus A$ for $A \subseteq N$. Observe that elementary imsets are closed under (the composition with) reflection. Thus, the cone $\mathcal{R}(N)$ and its Hilbert basis $\mathcal{H}(N)$ are closed under the reflection, too. In particular, some of 35 permutation equivalence classes are reflections of others. If the reflection is taken into consideration, the number of resulting equivalence classes in $\mathcal{H}(N)$ is lower, namely 21. Their representatives are given in Table 6.

$\emptyset$	0	0	2	2	1	1	0	0	1	0	0	1	0	2	0	0	1	2	3	3	2
$\{a\}$	0	0	0	0	0	0	1	1	0	1	0	1	1	-1	1	1	0	0	-1	1	1
$\{b\}$	0	0	0	0	0	1	1	1	0	1	1	0	0	0	1	1	0	0	-1	-1	0
$\{c\}$	0	0	0	0	1	1	0	0	1	1	1	1	1	0	1	1	1	0	1	-1	0
$\{d\}$	0	0	0	-2	0	0	1	1	0	1	1	0	1	-1	1	1	2	0	-1	-1	0
$\{e\}$	0	0	-2	0	0	-1	1	1	0	1	1	-1	1	-1	1	1	-1	-2	-2	-2	-2
$\{a,b\}$	0	0	-1	-1	0	-1	-1	-1	-1	-2	0	-1	0	0	-2	-1	-1	-1	0	-1	-1
$\{a,c\}$	0	0	-1	-1	-1	1	0	0	-1	-1	0	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
$\{a,d\}$	0	0	1	1	-1	-1	-2	-2	0	-1	0	-1	1	1	-1	-1	-1	1	0	-1	-1
$\{a,e\}$	0	0	1	0	2	1	1	0	2	1	0	0	-2	1	0	-1	1	2	1	2	2
$\{b,c\}$	0	0	0	0	-1	-2	0	0	-1	-1	-1	-1	0	-1	-1	-1	-1	1	-1	0	-1
$\{b,d\}$	0	0	-1	1	2	0	0	0	2	1	-1	1	0	1	0	-1	-1	-1	0	0	-1
$\{b,e\}$	0	0	1	-1	-1	0	-2	-2	0	-1	-1	1	0	1	-1	-1	1	1	1	1	1
$\{c,d\}$	0	0	-1	1	-1	-1	0	0	-1	-1	-1	-1	-2	1	-1	-1	-2	-1	-1	0	-1
$\{c,e\}$	0	0	1	-1	-1	0	0	0	-1	-1	-1	0	1	1	-1	-1	0	1	1	1	1
$\{d,e\}$	0	1	1	1	0	1	-1	-1	-1	-2	-1	1	-1	0	-2	-1	0	1	1	1	1
$\{a,b,c\}$	0	0	1	1	0	0	-1	-1	1	1	-1	1	-1	1	1	0	2	0	1	1	1
$\{a,b,d\}$	0	0	0	-1	-1	1	1	1	-1	0	-1	0	-2	-2	1	0	1	0	1	1	1
$\{a,b,e\}$	0	0	0	1	-1	-1	0	1	-1	0	1	-1	1	-1	1	2	0	-1	-1	-1	-1
$\{a,c,d\}$	0	0	0	0	2	0	1	1	1	1	1	1	0	-1	1	2	1	0	1	1	1
$\{a,c,e\}$	0	0	-1	1	-1	-2	-2	-1	-1	-1	-1	1	0	-2	0	0	-1	-1	0	-1	-1
$\{a,d,e\}$	0	0	-2	-1	-1	0	0	1	-1	0	-1	-1	0	0	1	0	-1	-2	-1	-1	-1
$\{b,c,d\}$	0	0	1	-1	-1	1	-1	-1	-1	-1	1	0	1	-1	0	1	1	0	1	1	2
$\{b,c,e\}$	0	0	-1	1	2	1	1	1	1	1	1	-1	-2	-1	1	1	-1	-2	0	-1	-1
$\{b,d,e\}$	0	-1	-1	0	-1	-1	1	1	-1	0	1	-1	-1	-1	1	1	-1	-1	-1	-1	-1
$\{c,d,e\}$	1	-1	0	-1	0	-1	-1	-1	1	1	1	-1	0	-1	1	1	1	-1	0	-1	-1
$\{a,b,c,d\}$	0	0	-1	0	0	-1	0	0	0	0	0	-1	1	1	-1	-1	-2	0	-2	-2	-2
$\{a,b,c,e\}$	0	0	0	-2	0	1	1	0	0	0	0	0	1	1	-1	-1	0	1	-1	0	0
$\{a,b,d,e\}$	0	0	1	0	1	0	-1	-2	1	0	0	1	1	1	-2	-1	0	1	1	0	0
$\{a,c,d,e\}$	-1	0	1	0	0	1	1	0	0	0	0	0	0	1	-1	-1	0	1	-1	0	0
$\{b,c,d,e\}$	-1	1	0	0	0	0	0	0	0	0	-2	1	1	1	-1	-2	0	2	-1	1	1
$\{a,b,c,d,e\}$	1	0	1	2	1	1	1	2	1	1	2	1	0	0	3	3	2	0	3	2	2

Table 6: The 21 columns correspond to representatives of equivalence classes of $\mathcal{H}(N)$.

The consequence of finding $\mathcal{H}(N)$ for $|N| = 5$ in Bruns et al. (2010) is as follows:

Proposition 6 For $|N| = 5$, the least factor $n_* \in \mathbb{N}$ such that, for any imset u over N, it holds $u \in \mathcal{S}(N) \Leftrightarrow n_* \cdot u \in \mathcal{C}(N)$, is $n_* = 2$.

Proof Using a computer, we checked that every element $v \in \mathcal{H}(N)$ for $|N| = 5$ has the property that $2 \cdot v$ is a combinatorial imset. Now, given $u \in \mathcal{S}(N)$, consider its Hilbert basis decomposition $u = \sum_{v \in \mathcal{H}(N)} k_v \cdot v$, $k_v \in \mathbb{Z}_+$ and write $2 \cdot u = \sum_{v \in \mathcal{H}(N)} k_v \cdot (2 \cdot v)$, where each summand $2 \cdot v$ is known to be a combinatorial imset. In particular, $2 \cdot u \in \mathcal{C}(N)$. ■

Appendix C. Example of the Application of the LP Method

The aim of this text is to illustrate the LP method from Section 4.4 by an example. We assume that the reader is familiar with the details of the tableau form of the simplex method (see, for example, Schrijver, 1986, § 11.2). Consider the same situation as in Example 1:

$$N = \{a,b,c\}, \quad L = \{\, a \perp\!\!\!\perp b \,|\, c\,,\ a \perp\!\!\!\perp c \,|\, \emptyset\,\}, \ \text{ and } \ t : a \perp\!\!\!\perp b \,|\, \emptyset\,,$$

which leads to the task to verify/disprove $u \rightharpoonup v$ for

$$u = u_L = \delta_{abc} - \delta_{bc} - \delta_a + \delta_\emptyset \ \text{ and } \ v = u_t = \delta_{ab} - \delta_a - \delta_b + \delta_\emptyset\,.$$

By Lemma 5, this is equivalent to solving the system of equations (9). In this case, it means searching for a non-negative solution $\mathbf{x} \geq \mathtt{0}$ of $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A}$ is 8×7-matrix and $\mathbf{b}$ a column 8-vector specified as follows:

A	$\lambda_{\langle a,b\mid c\rangle}$	$\lambda_{\langle a,c\mid b\rangle}$	$\lambda_{\langle b,c\mid a\rangle}$	$\lambda_{\langle a,b\mid\emptyset\rangle}$	$\lambda_{\langle a,c\mid\emptyset\rangle}$	$\lambda_{\langle b,c\mid\emptyset\rangle}$	k	**b**
$\emptyset$	0	0	0	+1	+1	+1	−1	−1
a	0	0	+1	−1	−1	0	+1	+1
b	0	+1	0	−1	0	−1	0	+1
c	+1	0	0	0	−1	−1	0	0
ab	0	−1	−1	+1	0	0	0	−1
ac	−1	0	−1	0	+1	0	0	0
bc	−1	−1	0	0	0	+1	+1	0
abc	+1	+1	+1	0	0	0	−1	0

Here, the first 6 columns of $\mathbf{A}$ are elementary imsets, the 7-th column is $-u$, while $\mathbf{b}$ encodes $-v$. Moreover, the columns of $\mathbf{A}$ are named by the respective (searched) non-negative coefficients, while the rows are named by corresponding subsets of N. To get the standard case $\mathbf{b} \geq \mathtt{0}$ we modify it by multiplying by (-1) the rows of $\mathbf{A}$ and components of $\mathbf{b}$ corresponding $\emptyset$ and ab. The modified system is $\mathbf{A}'\mathbf{x} = \mathbf{b}'$, where one has:

$\mathbf{A}'$	$\lambda_{\langle a,b\mid c\rangle}$	$\lambda_{\langle a,c\mid b\rangle}$	$\lambda_{\langle b,c\mid a\rangle}$	$\lambda_{\langle a,b\mid\emptyset\rangle}$	$\lambda_{\langle a,c\mid\emptyset\rangle}$	$\lambda_{\langle b,c\mid\emptyset\rangle}$	k	$\mathbf{b}'$
$\emptyset$	0	0	0	−1	−1	−1	+1	+1
a	0	0	+1	−1	−1	0	+1	+1
b	0	+1	0	−1	0	−1	0	+1
c	+1	0	0	0	−1	−1	0	0
ab	0	+1	+1	−1	0	0	0	+1
ac	−1	0	−1	0	+1	0	0	0
bc	−1	−1	0	0	0	+1	+1	0
abc	+1	+1	+1	0	0	0	−1	0

We are going to solve the auxiliary LP problem (4) (see Section 3.4), to which purpose we create a tableau with additional identity submatrix and cost vector.

	$\lambda_{\langle a,b\mid c\rangle}$	$\lambda_{\langle a,c\mid b\rangle}$	$\lambda_{\langle b,c\mid a\rangle}$	$\lambda_{\langle a,b\mid\emptyset\rangle}$	$\lambda_{\langle a,c\mid\emptyset\rangle}$	$\lambda_{\langle b,c\mid\emptyset\rangle}$	k	$\emptyset$	a	b	c	ab	ac	bc	abc	
cost	0	0	0	0	0	0	0	+1	+1	+1	+1	+1	+1	+1	+1	0
$\emptyset$	0	0	0	−1	−1	−1	+1	+1	0	0	0	0	0	0	0	+1
a	0	0	+1	−1	−1	0	+1	0	+1	0	0	0	0	0	0	+1
b	0	+1	0	−1	0	−1	0	0	0	+1	0	0	0	0	0	+1
c	+1	0	0	0	−1	−1	0	0	0	0	+1	0	0	0	0	0
ab	0	+1	+1	−1	0	0	0	0	0	0	0	+1	0	0	0	+1
ac	−1	0	−1	0	+1	0	0	0	0	0	0	0	+1	0	0	0
bc	−1	−1	0	0	0	+1	+1	0	0	0	0	0	0	+1	0	0
abc	+1	+1	+1	0	0	0	−1	0	0	0	0	0	0	0	+1	0

The next step is to "recompute" the cost vector to get zeros above the identity submatrix. In our case it means subtracting the sum of 8 rows that correspond to sets from the previous version of the cost vector. In the additional row, bullets indicate the current basis and the corresponding (starting) vertex is given by the coefficients indicated.

	$\lambda_{\langle a,b\|c\rangle}$	$\lambda_{\langle a,c\|b\rangle}$	$\lambda_{\langle b,c\|a\rangle}$	$\lambda_{\langle a,b\|\emptyset\rangle}$	$\lambda_{\langle a,c\|\emptyset\rangle}$	$\lambda_{\langle b,c\|\emptyset\rangle}$	k	$\emptyset$	a	b	c	ab	ac	bc	abc	
cost	0	−2	−2	+4	+2	+2	−2	0	0	0	0	0	0	0	0	−4
$\emptyset$	0	0	0	−1	−1	−1	+1	+1	0	0	0	0	0	0	0	+1
a	0	0	+1	−1	−1	0	+1	0	+1	0	0	0	0	0	0	+1
b	0	+1	0	−1	0	−1	0	0	0	+1	0	0	0	0	0	+1
c	+1	0	0	0	−1	−1	0	0	0	0	+1	0	0	0	0	0
ab	0	+1	+1	−1	0	0	0	0	0	0	0	+1	0	0	0	+1
ac	−1	0	−1	0	+1	0	0	0	0	0	0	0	+1	0	0	0
bc	−1	−1	0	0	0	+1	+1	0	0	0	0	0	0	+1	0	0
abc	+1	+1	+1	0	0	0	−1	0	0	0	0	0	0	0	+1	0
								•	•	•	•	•	•	•	•	
vert								1	1	1	0	1	0	0	0	

Now, the proper algorithm can start. The strategy for the choice of the pivoting position from Schrijver (1986, § 11.2) leads to pivoting on $abc \times \lambda_{\langle a,c|b\rangle}$-cell, which results in the following tableau:

	$\lambda_{\langle a,b\|c\rangle}$	$\lambda_{\langle a,c\|b\rangle}$	$\lambda_{\langle b,c\|a\rangle}$	$\lambda_{\langle a,b\|\emptyset\rangle}$	$\lambda_{\langle a,c\|\emptyset\rangle}$	$\lambda_{\langle b,c\|\emptyset\rangle}$	k	$\emptyset$	a	b	c	ab	ac	bc	abc	
cost	+2	0	0	+4	+2	+2	−4	0	0	0	0	0	0	0	+2	−4
$\emptyset$	0	0	0	−1	−1	−1	+1	+1	0	0	0	0	0	0	0	+1
a	0	0	+1	−1	−1	0	+1	0	+1	0	0	0	0	0	0	+1
b	−1	0	−1	−1	0	−1	+1	0	0	+1	0	0	0	0	−1	+1
c	+1	0	0	0	−1	−1	0	0	0	0	+1	0	0	0	0	0
ab	−1	0	0	−1	0	0	+1	0	0	0	0	+1	0	0	−1	+1
ac	−1	0	−1	0	+1	0	0	0	0	0	0	0	+1	0	0	0
bc	0	0	+1	0	0	+1	0	0	0	0	0	0	0	+1	+1	0
abc	+1	+1	+1	0	0	0	−1	0	0	0	0	0	0	0	+1	0
		•						•	•	•	•	•	•	•		
vert		0						1	1	1	0	1	0	0		

The vertex and the value of the cost function in it remain unchanged, just the vertex is expresses using another basis. The same procedure now leads to the choice of another pivoting cell: the column corresponds to k and the row to $\emptyset$.

	$\lambda_{\langle a,b\|c\rangle}$	$\lambda_{\langle a,c\|b\rangle}$	$\lambda_{\langle b,c\|a\rangle}$	$\lambda_{\langle a,b\|\emptyset\rangle}$	$\lambda_{\langle a,c\|\emptyset\rangle}$	$\lambda_{\langle b,c\|\emptyset\rangle}$	k	$\emptyset$	a	b	c	ab	ac	bc	abc	
cost	+2	0	0	0	−2	−2	0	+4	0	0	0	0	0	0	+2	0
$\emptyset$	0	0	0	−1	−1	−1	+1	+1	0	0	0	0	0	0	0	+1
a	0	0	+1	0	0	+1	0	−1	+1	0	0	0	0	0	0	0
b	−1	0	−1	0	+1	0	0	−1	0	+1	0	0	0	0	−1	0
c	+1	0	0	0	−1	−1	0	0	0	0	+1	0	0	0	0	0
ab	−1	0	0	0	+1	+1	0	−1	0	0	0	+1	0	0	−1	0
ac	−1	0	−1	0	+1	0	0	0	0	0	0	0	+1	0	0	0
bc	0	0	+1	0	0	+1	0	0	0	0	0	0	0	+1	+1	0
abc	+1	+1	+1	−1	−1	−1	0	+1	0	0	0	0	0	0	+1	+1
		•					•		•	•	•	•	•	•		
vert		1					1		0	0	0	0	0	0		

Now, we observe the decrease in the value of the cost function: the corresponding vertex is given by $k = 1$, $\lambda_{\langle a,c|b\rangle} = 1$ and remaining coefficients vanishing. The value of the cost function is already 0, a further terminating iteration (= pivoting on $b \times \lambda_{\langle a,c|\emptyset\rangle}$-cell) does not change the vertex and, hence, the value of the cost function in it. Thus, we have successfully found a non-negative solution to (9). In particular, we confirmed $u_L \rightharpoonup u_t$ by the LP method.

Appendix D. Decomposition Interpretation of the LP Method

In spite of the different formulations, the procedure from Section 4.2.1 and the LP approach from Section 4.4 have common points. Given $u \in \mathcal{C}(N)$ and $v \in \mathcal{E}(N)$, both methods try to find a decomposition of $k \cdot u - v$ into elementary imsets. A formal difference is that k and the searched coefficients are integers in Section 4.2.1 but reals in Section 4.4. We try now to explain the analogy to the reader familiar with the details of the simplex method.

To see the analogy let us assume that k is a fixed natural number, as in Section 4.2.1. For example, let k be the upper limit for the factor l in (6). This ensures that $k \cdot u - v \in \mathcal{C}(N)$ iff $u \rightharpoonup v$. Put $b(S) = k \cdot u(S) - v(S)$ for $S \subseteq N$ and search for a non-negative solution in λ_w, $w \in \mathcal{E}(N)$ to the system

$$\sum_{w \in \mathcal{E}(N)} \lambda_w \cdot w(S) = b(S) \qquad \text{for any } S \subseteq N.$$

The set of such solutions is a polyhedron $\hat{\mathsf{Q}} = \{\mathbf{x} \in \mathbb{R}^n; \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathtt{0}\}$, with $\mathbf{A} \in \mathbb{R}^{d \times n}$ and $\mathbf{b} \in \mathbb{R}^d$. Here, the rows of $\mathbf{A}$ correspond to subsets of N ($d = |\mathcal{P}(N)|$), the columns to elementary imsets ($n = |\mathcal{E}(N)|$) and $\mathbf{b}$ has $b(S)$, $S \subseteq N$ as components. The only difference from the situation in Section 4.4 is that we have one column less.

As mentioned in Section 3.4, $\mathbf{b}$ (and $\mathbf{A}$) can be modified so that it has non-negative integers as components and entries in $\mathbf{A}$ remain in $\{-1, 0, 1\}$. The simplex method takes $(\mathtt{0}, \mathbf{b}) \equiv (\mathbf{x}_0, \mathbf{z}_0)$ as its starting iteration and generates a sequence $(\mathbf{x}_0, \mathbf{z}_0), \ldots, (\mathbf{x}_m, \mathbf{z}_m)$, $m \geq 0$ of points in the feasible region of (4). The final iteration $(\mathbf{x}_m, \mathbf{z}_m)$ then minimizes the function $(\mathbf{x}, \mathbf{z}) \mapsto \langle \mathtt{1}, \mathbf{z} \rangle$ over that region. As explained in Section 3.4, $\hat{\mathsf{Q}} \neq \emptyset$ iff $\mathbf{z}_m = \mathtt{0}$.

Now, the move from $(\mathtt{0}, \mathbf{b}) \equiv (\mathbf{x}_0, \mathbf{z}_0)$ to $(\mathbf{x}_1, \mathbf{z}_1)$ means the following. One finds $w \in \mathcal{E}(N)$, corresponding to a (column) component x_w, and a set $S' \subseteq N$, corresponding to a component $z_{S'}$, such that $b(S') - \lambda_w \cdot w(S') = 0$ with $\lambda_w \geq 0$. The components of $\mathbf{z}_1$ are then $b(S) - \lambda_w \cdot w(S)$ for $S \subseteq N$. Since $\mathbf{b} = \mathbf{z}_0$ has integral components, $\lambda_w \in \mathbb{Z}^+$. In other words, the first step of the simplex method, the move from $\mathbf{z}_0$ to $\mathbf{z}_1$, means subtracting a non-negative integral multiple of an elementary imset w from $\mathbf{b} = \mathbf{z}_0$ so that the result remains a non-negative vector.

Although further steps already cannot be interpreted as elementary imset subtraction, the following still holds. If $\hat{\mathsf{Q}} \neq \emptyset$, the simplex method finds iteratively the decomposition of $\mathbf{b}$ into elementary imsets with non-negative coefficients. Moreover, in every iteration, one set (= a component of $\mathbf{z}$) is "vanished". This is analogous to the procedure from Section 4.2.1, the difference is that the heuristic for finding the next set S' to be "vanished" is given by the pivoting operation.

References

Remco R. Bouckaert and Milan Studený. Racing algorithms for conditional independence inference. *International Journal of Approximate Reasoning*, 45(2):386–401, 2007.

Winfried Bruns, Bogdan Ichim, and Christof Söger. NORMALIZ, a tool for computations in affine monoids, vector configurations, lattice polytopes and rational cones. Available electronicly at `http://www.math.uos.de/normaliz/`, 2009.

Winfried Bruns, Raymond Hemmecke, Bogdan Ichim, Matthias Köppe, and Christof Söger. Challenging computations of Hilbert bases of cones associated with algebraic statistics. To appear in *Experimental Mathematics*, e-print available at `arxiv.org/abs/1001.4145v1`, 2010.

David M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(2):507–554, 2002.

Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer Verlag, 1999.

A. Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society B*, 41(1):1–31, 1979.

Matthias Franz. CONVEX, a Maple package for convex geometry. Available electronicly at `www.math.uwo.ca/~mfranz/convex/`, 2009.

Komei Fukuda. CDD and CDD+, an implementation of the double description method. Available electronicly at `www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html`, 2008.

Dan Geiger, Tom Verma, and Judea Pearl. D-separation: from theorems to algorithms. In *Proceedings of the 5th Conference on Uncertainty in Artificial Intelligence*, pages 118–125, Elsevier, 1989.

Raymond Hemmecke, Jason Morton, Anne Shiu, Bernd Sturmfels, and Oliver Wienand. Three counter-examples on semi-graphoids. *Combinatorics, Probability and Computing*, 17(2):239–257, 2008.

IBM Ilog team. CPLEX, a mathematical programming optimizer. Available electronicly at `www-01.ibm.com/software/integration/optimization/cplex/`, 2009.

Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer Verlag, 2001.

Steffen L. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

Richard E. Neapolitan. *Learning Bayesian Networks*. Pearson Prentice Hall, 2004.

Mathias Niepert, Dirk van Gucht, and Marc Gyssens. On the conditional independence implication problem, a lattice theoretic approach. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 435–443, AUAI Press, 2008.

Mathias Niepert. Logical inference algorithms and matrix representations for probabilistic conditional independence. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 428–435, AUAI Press, 2009.

Mathias Niepert, Dirk van Gucht, and Marc Gyssens. Logical and algorithmic properties of stable conditional independence. *International Journal of Approximate Reasoning*, 51(5):531–543, 2010.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley, 1986.

Petr Šimeček. *Independence models* (in Czech). PhD thesis, Charles University, Prague, 2007.

Milan Studený. Multiinformation and the problem of characterization of conditional independence relations. *Problems of Control and Information Theory*, 18(1):3–16, 1989.

Milan Studený. Convex set functions I and II. Research reports n. 1733 and 1734, Institute of Information Theory and Automation, Prague, November 1991.

Milan Studený. Conditional independence relations have no finite complete characterization. In *Information Theory, Statistical Decision Functions and Random Processes, Transactions of the 11th Prague Conference, vol. B* (S. Kubík, J. Á. Víšek eds.), pages 377–396, Kluwer, 1992.

Milan Studený. Convex cones in finite-dimensional real vector spaces. *Kybernetika*, 29(2):180–200, 1993.

Milan Studený, Remco R. Bouckaert, and Tomáš Kočka. Extreme supermodular set functions over five variables. Research report n. 1977, Institute of Information Theory and Automation, Prague, January 2000.

Milan Studený. Structural imsets, an algebraic method for describing conditional independence structures. In *Proceedings of IPMU 2004* (B. Bouchon-Meunier, G. Coletti, R. R. Yager eds.), pages 1323–1330, 2004.

Milan Studený. *Probabilistic Conditional Independence Structures*. Springer Verlag, 2005.

Milan Studený, Jiří Vomlel, and Raymond Hemmecke. A geometric view on learning Bayesian network structures. *International Journal of Approximate Reasoning*, 51(5):573–586, 2010.

Milan Studený and Jiří Vomlel. On open questions in the geometric approach to structural learning Bayesian nets. Accepted in *International Journal of Approximate Reasoning*, to appear in 2011.

4ti2 team. 4TI2, a software package for algebraic, geometric and combinatorial problems on linear spaces. Available electronicly at `www.4ti2.de`, 2008.

Journal of Machine Learning Research 11 (2010) 3481-3518 Submitted 6/09; Revised 8/10; Published 12/10

An Exponential Model for Infinite Rankings

Marina Meilă MMP@STAT.WASHINGTON.EDU
Le Bao LEBAO@STAT.WASHINGTON.EDU
Department of Statistics
University of Washington
Seattle, WA 98195-4322, USA

Editor: David Blei

Abstract

This paper presents a statistical model for expressing preferences through rankings, when the number of alternatives (items to rank) is large. A human ranker will then typically rank only the most preferred items, and may not even examine the whole set of items, or know how many they are. Similarly, a user presented with the ranked output of a search engine, will only consider the highest ranked items. We model such situations by introducing a stagewise ranking model that operates with finite ordered lists called top-t orderings over an infinite space of items. We give algorithms to estimate this model from data, and demonstrate that it has sufficient statistics, being thus an exponential family model with continuous and discrete parameters. We describe its conjugate prior and other statistical properties. Then, we extend the estimation problem to multimodal data by introducing an *Exponential-Blurring-Mean-Shift* nonparametric clustering algorithm. The experiments highlight the properties of our model and demonstrate that infinite models over permutations can be simple, elegant and practical.

Keywords: permutations, partial orderings, Mallows model, distance based ranking model, exponential family, non-parametric clustering, branch-and-bound

1. Introduction

The stagewise ranking model of Fligner and Verducci (1986), also known as *generalized Mallows (GM)*, has been recognized as particularly appropriate for modeling the human process of ranking. This model assigns a permutation π over n items a probability that decays exponentially with its distance to a *central permutation* σ. Here we study this class of models in the limit $n \to \infty$, with the assumption that out of the infinitely many items ordered, one only observes those occupying the first t ranks.

Ordering an infinite number of items is common in retrieval tasks: search engines, programs that match a face, or a fingerprint, or a biological sequence against a database, all output the first t items in a ordering over virtually infinitely many objects. We shall call this output a *top-t* ordering. Unlike machines, people can only reliably rank a small number of items. The GM model has been successfully used to model human ranking decisions. We can view the difference between the standard GM model and the *Infinite GM model* that we introduce here as the difference between an election where each voter returns an ordering of a small number of preselected candidates (nominees) and a "grassroots" election process, where everyone can nominate and order their own favourites from a virtually unlimited population. For instance, the difference between "Order the following issues by how much you care about them" vs. "List in order the issues that you care most about" illustrates

the difference between the standard and the Infinite GM models. By these examples, we argue that the Infinite GM corresponds to realistic scenarios. An even more realistic scenario, that we will not tackle for now, is one where a voter (ranker) does not have access to the whole population of items (e.g., a search engine only orders a subset of the web, or a reviewer only evaluates a subset of the submissions to a conference).

After defining the infinite GM model, we show that it has sufficient statistics and give algorithms for estimating its parameters from data in the Maximum Likelihood (ML) framework. To be noted that our model will have an infinite number of parameters, of which only a finite number will be constrained by the data from any finite sample.

Then, we consider the non-parametric clustering of top-t ranking data. Non-parametric clustering is motivated by the fact that in many real applications the number of clusters is not known and outliers are possible. Outliers are known to throw off estimation in an exponential model, unless the tails are very heavy. We introduce an adapted version of the well known Gaussian Blurring Mean-Shift algorithm (Carreira-Perpiñán, 2006) (GBMS) that we call exponential blurring Mean Shift (EBMS).

2. The Infinite Generalized Mallows Model

In this section, we give definitions of key terms used in the article and introduce the *Infinite Generalized Mallows* (IGM) model.

2.1 Permutations, Infinite Permutations and top-t Orderings

A permutation σ is a function from a set of *items* $\{i_1, i_2, \ldots i_n\}$ to the set of *ranks* $1:n$. W.l.o.g. the set of items can be considered to be the set $1:n$. Therefore $\sigma(i)$ denotes the *rank* of item i and $\sigma^{-1}(j)$ denotes the item at rank j in σ.

There are many other ways to represent permutations, of which we will use three, the *ranked list*, the *matrix* and the *inversion table* representation; all three will be defined shortly.

In this paper, we consider permutations over a countable set of items, assumed for convenience to be the the set of positive natural numbers $\mathbb{P} = \{1, 2, \ldots, i \ldots\}$. It is easy to see that the notations $\sigma(i), \sigma^{-1}(j)$ extend immediately to countable items. This will be the case with the other definitions; hence, from now on, we will always consider that the set of items is $\mathbb{P}$.

Any permutation σ can be represented by the (infinite) *ranked list* $(\sigma^{-1}(1)|\sigma^{-1}(2)|\ldots|\sigma^{-1}(j)|\ldots)$. For example, let $\sigma = (2|3|1|5|6|4|\ldots|3n-1|3n|3n-2|\ldots)$. Then $\sigma(1) = 3$ means that item 1 has rank 3 in this permutation; $\sigma(2) = 1$ means that item 2 has rank 1, etc. Conversely, $\sigma^{-1}(1) = 2$ and $\sigma^{-1}(3j) = 3j-2$ mean that the first in the list representation of σ is item 2, and that at rank $3j$ is to be found item $3j-2$, respectively. Often we will call the list representation of a permutation an *ordering*.

A top-t ordering π is the prefix $(\pi^{-1}(1)|\pi^{-1}(2)|\ldots|\pi^{-1}(t))$ of some infinite ordering. For instance, the top-3 ordering of the above σ is $(2|3|1)$.

A top-t ordering can be seen as defining a set consisting of those infinite orderings which start with the prefix π. If we denote by $\mathcal{S}_{\mathbb{P}}$ the set of all permutations over $\mathbb{P}$ and by $\mathcal{S}_{\mathbb{P}-t} = \{\sigma \in \mathcal{S}_{\mathbb{P}} \,|\, \sigma(i) = i, \text{for} \, i = 1:t\}$ the subgroup of all permutations that leave the top-t ranks unchanged, then a top-t ordering π corresponds to a unique element of the right coset $\mathcal{S}_{\mathbb{P}}/\mathcal{S}_{\mathbb{P}-t}$.

We will use Greek letters like π and σ for both full permutations and for top-t orderings to keep the notation light. But we will distinguish almost always between "central permutations"

ideal infinite objects denoted by σ, and observed orderings, denoted by π, which by virtue of being observed, are always top-t , that is, truncated. Hence, unless otherwise stated, π will denote a top-t ordering, while σ will denote an infinite permutation.

2.2 The Permutation Matrix Representation and the Inversion Table

Now we introduce the two other ways use to represent permutations and top-t orderings.

For any σ, the *permutation matrix* Σ corresponding to σ has $\Sigma_{ij} = 1$ iff $\sigma(i) = j$ and $\Sigma_{ij} = 0$ otherwise. If σ is an infinite permutation, Σ will be an infinite matrix with exactly one 1 in every row and column. For two permutations σ, σ' over $\mathbb{P}$, the matrix product $\Sigma\Sigma'$ corresponds to the function composition $\sigma' \circ \sigma$.

The matrix Π of a top-t ordering π is a truncation of some infinite permutation matrix Σ. It has t columns, each with a single 1 in $\pi^{-1}(j)$, for $j = 1:t$.

For example, if $\sigma = (2|3|1|7|4|\ldots)$ and $\pi = (2|3|1)$ is its top-3 ordering, then

$$\Sigma = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & \ldots \\ 1 & 0 & 0 & 0 & 0 & \ldots \\ 0 & 1 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix} \quad \text{and} \quad \Pi = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ \ldots & \ldots & \ldots \end{bmatrix}. \tag{1}$$

For a permutation σ and a top-t ordering π, the matrix $\Sigma^T\Pi$ corresponds to the list of ranks in σ of the items in π. In this context, one can consider σ as a one-to-one relabeling of the set $\mathbb{P}$.

The *inversion table* of a permutation σ, with respect to the identity permutation id is an infinite sequence of non-negative integers $(s_1, s_2, \ldots)$ which are best defined algorithmically and recursively. We consider the ranked list $(1|2|3|\ldots)$. In it, we find $\sigma^{-1}(1)$ the first item of σ, and count how many positions past the head of the list it lies. This is s_1, and it always equals $\sigma^{-1}(1) - 1$. Then we delete the entry $\sigma^{-1}(1)$ from the list, and look up $\sigma^{-1}(2)$; s_2 is the number of positions past the head of this list where we find $\sigma^{-1}(2)$. We then delete $\sigma^{-1}(2)$ as well and proceed to find $\sigma^{-1}(3)$, which will give us s_3, etc. By induction, it follows that an infinite permutation can be represented uniquely by the list $(s_1, s_2, \ldots)$. Hence $s_j \in \{0, 1, 2, \ldots\}$, and, denoting by $1_{[p]}$ the function which is 1 if the predicate p is true and is 0 otherwise, we have

$$s_j(\sigma) \;=\; \sigma^{-1}(j) - 1 - \sum_{j'<j} 1_{[\sigma^{-1}(j') < \sigma^{-1}(j)]}.$$

It is also easy to see that, if π is a top-t ordering, it can be uniquely represented by an inversion table of the form $(s_1, \ldots s_t)$. If π is the top-t ordering of an infinite permutation σ, then the inversion table of π is the t-prefix of the inversion table of σ. This property makes the inversion table particularly convenient for our purposes.

For example, if $\sigma = (2|3|1|7|4|\ldots)$ and $\pi = (2|3|1)$, then $s_1(\sigma) = s_1(\pi) = 1$, $s_2(\sigma) = s_2(\pi) = 1$, $s_3(\sigma) = s_3(\pi) = 0$, $s_4(\sigma) = 3$, $s_5(\sigma) = 0$, etc.

The inversion table has a particularly simple interpretation in the matrix representation of σ or π: s_1 equals the number of zeros preceding 1 in column 1; we delete the row containing this 1, then count the number of zeros in column 2 preceding the 1 to obtain s_2; we delete the row containing this 1, then go to column 3 to count the zeros preceding the 1 in column 3 in order to obtain s_3, and so on. The reader can verify that $s_{1:3}(\pi) = (1, 1, 0)$ from the matrix Π in Equation (1) above.

Another property of the inversion table is that it can be defined with respect to any infinite permutation σ_0, by letting the ordered list corresponding to σ_0 replace the list $(1|2|3|\ldots)$ in the above definition of the inversion table as follows:

$$s_j(\sigma|\sigma_0) \;=\; \sigma_0(\sigma^{-1}(j)) - 1 - \sum_{j'<j} 1_{[\sigma_0(\sigma^{-1}(j'))<\sigma_0(\sigma^{-1}(j))]}. \tag{2}$$

In other words, $1+s_j$ is the rank of $\sigma^{-1}(j)$ in $\sigma_0\big|_{\mathbb{P}\setminus\{\sigma^{-1}(1),\ldots\sigma^{-1}(j-1)\}}$.

In matrix representation, $s_1(\sigma|\sigma_0)$ is the number of 0's preceding the 1 in the first column of $\Sigma_0^T\Sigma$; after we delete the row containing this 1, $s_2(\sigma|\sigma_0)$ is the number of 0's preceding the 1 in the second column, and so on. For a top-t ordering π, this operation is done on the matrix $\Sigma_0^T\Pi$.

For example, for $\pi = (3|2|1)$ and $\sigma_0 = (3|4|2|1|\ldots)$ the matrix representation is

$$\Sigma_0^T\Pi \;=\; \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \ldots & \ldots & \ldots \end{bmatrix},$$

and $s_1(\pi|\sigma_0) = 0$, $s_2(\pi|\sigma_0) = 1$, $s_3(\pi|\sigma_0) = 1$.

If σ_0 is given, π is completely determined by the inversion table $s_{1:t}(\pi|\sigma)$. Equation (2) can be interpreted as a recursive algorithm to construct π from σ, which we briefly describe here. We imagine σ to be an ordered list of available items. From it, we choose the first rank in π by skipping the first s_1 ranks in σ and picking $\pi^{-1}(1) = \sigma^{-1}(s_1+1)$. Once $\pi^{-1}(1)$ is picked, this item is deleted from the ordering σ. From this new list of available items, the second rank in π is picked by skipping the first s_2 ranks, and chosing the item in the (s_2+1)-th rank. This item is also deleted, and one proceeds to choose $\pi^{-1}(3), \pi^{-1}(4), \ldots \pi^{-1}(t)$, etc. in a similar manner. This reconstruction algorithm proves that the representation $(s_{1:t})$ uniquely determines π. It is also easy to see that if π is a prefix of σ, that is, if $\pi^{-1}(j) = \sigma^{-1}(j)$ for $j = 1:t$, then $s_1 = s_2 = \ldots = s_t = 0$.

For an example we now show how to reconstruct $\pi = (3|2|1)$ using $\sigma = (3|4|2|1|\ldots)$. Recall that the inversion table of π is given by $s_1(\pi|\sigma) = 0$, $s_2(\pi|\sigma) = 1$, $s_3(\pi|\sigma) = 1$.

Stage	π	σ	Comments
Initial	$()$	$(3\|4\|2\|1\|\ldots)$	
$j=1$	(3)	$(\not{3}\|4\|2\|1\|\ldots)$	Skip $s_1=0$ ranks from σ, then assign the current item to $\pi^{-1}(1)$
$j=2$	$(3\|2)$	$(\not{3}\|4\|\not{2}\|1\|\ldots)$	Skip $s_2=1$ ranks from σ, then assign the current item to $\pi^{-1}(2)$
$j=3$	$(3\|2\|1)$	$(\not{3}\|4\|\not{2}\|\not{1}\|\ldots)$	Skip $s_3=1$ ranks from σ, then assign the current item to $\pi^{-1}(3)$

The definition of the inversion table s is identical to the first equation of Section 3 from Fligner and Verducci (1986). A reciprocal definition of the inversion table is used by Meilă et al. (2007) and Stanley (1997) and is typically denoted by $(V_1, V_2, \ldots)$. The "V" form of the inversion table is closely related to the inversion table we use here. We discuss this relationship in Section 7.

2.3 Kendall Type Divergences

For finite permutations of n items π and σ,

$$d_K(\pi,\sigma) \;=\; \sum_{j=1}^{n-1} s_j(\pi|\sigma) \;=\; \sum_{j=1}^{n-1} s_j(\sigma|\pi)$$

denotes the *Kendall distance* (Mallows, 1957) (or *inversion distance*) which is a metric. In the above, the index j runs only to $n-1$ because for a finite permutation, $s_n \equiv 0$. The Kendall distance represents the number of adjacent transpositions needed to turn π into σ. An extension of the Kendall distance which has been found very useful for modeling purposes was introduced by Fligner and Verducci (1986). It is

$$d_\theta(\pi,\sigma) \;=\; \sum_{j=1}^{n-1} \theta_j s_j(\pi|\sigma),$$

with $\theta = (\theta_1, \ldots \theta_{n-1})$ a vector of real parameters, typically non-negative. Note that d_θ is in general not symmetric, nor does it always satisfy the triangle inequality.

For the case of countable items, we introduce the divergence

$$d_\theta(\pi,\sigma) \;=\; \sum_{j=1}^{t} \theta_j s_j(\pi|\sigma), \tag{3}$$

where π is a top-t ordering, σ is a permutation in $\mathcal{S}_\mathbb{P}$, and $\theta = (\theta_{1:t})$ a vector of *strictly positive* parameters.[1]

When θ_j are all equal $d_\theta(\pi,\sigma)$ is proportional to the Kendall distance between σ and the set of orderings compatible with π and counts the number of inversions needed to make σ compatible with π. In general, this "distance" between a top-t ordering and an infinite ordering is a *set distance*.[2]

2.4 A Probability Model over top-t Rankings of Infinite Permutations

Now we are ready to introduce the *Infinite Generalized Mallows* (IGM) model. We start with the observation that as any top-t ordering can be represented uniquely by a sequence of t natural numbers, defining a distribution over the former is equivalent to defining a distribution over the latter, which is a more intuitive task. In keeping with the GM paradigm of Fligner and Verducci (1986), each s_j is sampled independently from a discrete exponential with parameter $\theta_j > 0$.

$$P(s_j) \;=\; \frac{1}{\psi(\theta_j)} e^{-\theta_j s_j}, \qquad s_j = 0, 1, 2, \ldots. \tag{4}$$

The normalization constant ψ is

$$\psi(\theta_j) = \sum_{k=0}^{\infty} e^{-\theta_j k} = \frac{1}{1-e^{-\theta_j}}, \tag{5}$$

and the expectation of s_j is $E[s_j|\theta_j] = \frac{e^{-\theta_j}}{1-e^{-\theta_j}} = \frac{1}{e^{\theta_j}-1}$ the well known expectation of the discrete geometric distribution. Now we fix a permutation σ. Since any π is uniquely defined by σ and the inversion table $s_{1:t}(\pi|\sigma)$, Equations (4) and (5) define a distribution over top-t orderings, by $P_{\theta,\sigma}(\pi) = \prod_{j=1}^{t} P(s_j(\pi|\sigma))$. This is equivalent to

$$P_{\theta,\sigma}(\pi) \;=\; e^{-\sum_{j=1}^{t}[\theta_j s_j(\pi|\sigma) + \ln\psi(\theta_j)]}. \tag{6}$$

1. Definition 3 can be easily extended to a pair $\sigma, \sigma' \in \mathcal{S}_\mathbb{P}$, but in this case the divergence will often take infinite values.
2. A set distance, often called "distance" between two sets, is the minimum distance between elements in different sets, that is, $\delta(A,B) = \min_{x\in A, y\in B} \delta(x,y)$ for a metric or divergence δ. The set distance is not a metric, as it can easily be shown by counterexample.

The above $P_{\theta,\sigma}(\pi)$ has a t-dimensional real parameter θ and an infinite-dimensional discrete parameter σ. The normalization constant $\prod_{j=1}^{t}\psi(\theta_j)$ ensures that

$$\sum_{\pi\in\text{top-}t\text{ orderings of }\mathbb{P}} P_{\theta,\sigma}(\pi) \;=\; 1.$$

In contrast with the finite GM, the parameters θ_j must be strictly positive for the probability distribution to exist. The most probable π for any given t has $s_1(\pi|\sigma) = \ldots = s_t(\pi|\sigma) = 0$. This is the top-$t$ prefix of σ.

The permutation σ is called the *central permutation* of $P_{\theta,\sigma}$. The parameters θ control the spread around the mode σ. Larger θ_j correspond to more concentrated distributions. These facts are direct extensions of statements about the GM model from Fligner and Verducci (1986) and therefore the detailed proofs are omitted.

What is different about the IGM model definition w.r.t its finite counterpart is that the parameter σ is now an infinite sequence instead of a finite one. Another difference is the added condition that $\theta_j > 0$ which ensures that $\psi(\theta_j)$ is finite. This condition is not necessary in the finite case, which leads to the non-identifiability[3] of the parameter σ.

If $\theta_1 = \theta_2 = \ldots = \theta$ the IGM model is called a *single parameter* IGM model. In this case Equation (6) simplifies to

$$P_{\theta,\sigma}(\pi) \;=\; e^{-\theta\sum_{j=1}^{t} s_j(\pi|\sigma) - t\ln\psi(\theta)}.$$

2.5 The IGM Model is a Marginal Distribution

Any top-t ordering π stands for a set of infinite sequences starting with $s_{1:t}(\pi|\sigma)$. Therefore, $P_{\theta,\sigma}(\pi)$ can be viewed as the marginal of $s_{1:t}$ in the infinite product space defined by the distribution

$$P_{\theta,\sigma}(s) \;=\; e^{-\sum_{j=1}^{\infty}[\theta_j s_j + \ln\psi(\theta_j)]} \qquad s \in \mathbb{N}\times\mathbb{N}\times\ldots.$$

Because every infinite sequence s uniquely determines an infinite permutation, the distribution (6) also represents the probability of the π element of the right coset $\mathcal{S}_{\mathbb{P}}/\mathcal{S}_{\mathbb{P}-t}$, that is, the set of infinite permutations that have $(\pi^{-1}(1)|\pi^{-1}(2)|\ldots|\pi^{-1}(t))$ as a prefix. This fact was noted by Fligner and Verducci (1986) in the context of finite number of items. Thus, the IGM model (6) is the infinite counter part of the GM model.

Note also that the expected value of s_j is the mean of the geometric distribution $E[s_j] = \frac{1}{e^{\theta_j}-1} \equiv \xi_j$. Thus, the mean value parametrization of the IGM can be easily derived (proof omitted):

$$P_{\xi,\sigma}(\pi) \;=\; \prod_{j=1}^{t} \frac{\xi_j^{s_j(\pi|\sigma)}}{(\xi_j+1)^{s_j(\pi|\sigma)+1}}. \tag{7}$$

It is clear by now that the IGM $P_{\theta,\sigma}$ is an exponential family model over the sample space $(s_1, s_2, \ldots)$ (note that here σ plays no role). It is also evidently an exponential family model in θ over complete or top-t permutations. The next section will demonstrate that, less evidently, the IGM is in fact in the exponential family with respect to the discrete parameter σ as well.

3. The non-identifiability of the GM model is however not a severe problem for estimation, and can be removed by imposing $\theta_j \geq 0$ (Meilă et al., 2007).

3. Estimating the Model From Data

We are given a set of N top-t orderings $\mathcal{S}_N$. Each $\pi \in \mathcal{S}_N$ can have a different length t_π; all π are sampled independently from a $P_{\theta,\sigma}$ with unknown parameters. We propose to estimate θ, σ from this data in the ML paradigm. We will start by rewriting the log-likelihood of the model, in a way that will uncover a set of sufficient statistics. Then we will show how to estimate the model based on the sufficient statistics.

3.1 Sufficient Statistics

For any square (infinite) matrix $A \in \mathbb{R}^{\mathbb{P}\times\mathbb{P}}$, denote by $L(A) = \sum_{i>j} A_{ij}$ the sum of the elements below the diagonal of A. Let $L_\sigma(A) = L(\Sigma^T A \Sigma)$, and let $\mathbf{1}$ be a vector of all 1's. For any π, let t_π be its length and denote $t_{max} = \max_{\mathcal{S}_N} t_\pi$, $T = \sum_{\mathcal{S}_N} t_\pi$, and by n the number of distinct items observed in the data. As we shall see, t_{max} is the dimension of the concentration parameter θ, n is the order of the estimated central permutation σ, and T counts the total number of items in the data, playing a role akin to the sample size.

Proposition 1 *Let $\{N_j, q_j, Q_j\}_{j\geq 0}$ represent the following statistics: N_j is the number of $\pi \in \mathcal{S}_N$ that have length $t_\pi \geq j$ (in other words, that contain rank j); q_j is the vector $[q_{i,j}]_{i\in\mathbb{P}}$, with $q_{i,j}$ being the number of times i is observed in rank j in the data $\mathcal{S}_N$, $Q_j = [Q_{ii',j}]_{i',i\in\mathbb{P}}$ is a matrix whose element $Q_{ii',j}$ counts how many times $\pi(i) = j$ and $\pi(i') < j$. Then,*

$$\ln P_{\theta,\sigma}(\mathcal{S}_N) \;=\; -\sum_{j\geq 1} [\theta_j L_\sigma(R_j) + N_j \ln \psi(\theta_j)] \quad \text{with } R_j \;=\; q_j \mathbf{1}^T - Q_j, \tag{8}$$

To prove this result, we first introduce an alternative expression for the inversion table $s_j(\pi|\sigma)$. Let the data set $\mathcal{S}_N$ consist of a single permutation π and define $q_j(\pi), Q_j(\pi)$ and $R_j(\pi)$ similar to q_j, Q_j, R_j above. Then we have

Proposition 2

$$s_j(\pi|\sigma) \;=\; L_\sigma(q_j(\pi)\mathbf{1}^T - Q_j(\pi)). \tag{9}$$

Proof Let Q_0 be the infinite matrix that has 1 above the main diagonal and 0 elsewhere, $(Q_0)_{ij} = 1$ iff $j > i$ and let $\Pi_{:j}$ denote the j-th column of Π. It is then obvious that $L(A) = \text{trace}\,(Q_0 A)$ for any A.

By definition, s_j represents the number of 0's preceding 1 in column j, minus all the 1's in the submatrix $(\Sigma^T\Pi)_{1:\sigma(\pi^{-1}(j))-1,1:j-1}$. In other words,

$$\begin{aligned}
s_j(\pi|\sigma) &= \sum_{l\geq 1} (Q_0 \Sigma^T \Pi_{:j})_l (\mathbf{1} - \Sigma^T\Pi_{:1} - \Sigma^T\Pi_{:2} - \ldots \Sigma^T \Pi_{:j-1})_l, \\
&= (\mathbf{1} - \sum_{j'<j} \Sigma^T \Pi_{:j'})^T Q_0 \Sigma^T \Pi_{:j}, \\
&= \mathbf{1}^T Q_0 \Sigma^T \Pi_{:j} - \sum_{j'<j} \Pi_{:j'}^T \Sigma Q_0 \Sigma^T \Pi_{:j}, \\
&= \text{trace}\, Q_0 \Sigma^T \Pi_{:j} \mathbf{1}^T - \sum_{j'<j} \text{trace}\, \Pi_{:j} \Pi_{:j'}^T \Sigma Q_0 \Sigma^T, \\
&= \text{trace}\, Q_0 \Sigma^T [\Pi_{:j}\mathbf{1}^T - \sum_{j'<j} \Pi_{:j}\Pi_{:j'}^T \Sigma], \\
&= L(\Sigma^T [\Pi_{:j}\mathbf{1}^T - \sum_{j'<j} \Pi_{:j}\Pi_{:j'}^T]\Sigma) \;=\; L_\sigma(\Pi_{:j}\mathbf{1}^T - \sum_{j'<j} \Pi_{:j}\Pi_{:j'}^T).
\end{aligned}$$

We use the fact that multiplying left by Q_0 counts the zeros preceding 1 in a column in the first equality, traceAB = traceBA in the fourth and fifth equations, and the identity $\mathbf{1}^T\Sigma = \mathbf{1}^T$ in the last equation. We now note that $\Pi_{:j} = q_j(\pi)$ and $\sum_{j'<j}\Pi_{:j}\Pi_{:j'}^T = Q_j(\pi)$ and the result follows. □

Proof of Proposition 1. The log-likelihood of $\mathcal{S}_N$ is given by

$$\begin{aligned}\ln P_{\theta,\sigma}(\mathcal{S}_N) &= -\sum_{\pi\in\mathcal{S}_N}\left[\sum_{j=1}^{t}\pi\theta_j s_j(\pi|\sigma)+\ln\psi(\theta_j)\right],\\ &= -\sum_{j\geq 1}\left[\theta_j\sum_{\pi\in\mathcal{S}_N}s_j(\pi|\sigma)+N_j\ln\psi(\theta_j)\right].\end{aligned}$$

Because L_σ is a linear operator, the sum over $\pi\in\mathcal{S}_N$ equals

$$L_\sigma[(\sum_{\pi\in\mathcal{S}_N}q_j(\pi))\mathbf{1}^T-\sum_{\pi\in\mathcal{S}_N}Q_j(\pi))].$$

It is easy to verify now that the first sum represents q_j and the second one represents Q_j.
□

The sufficient statistics for the single parameter IGM model are described by the following corollary.

Corollary 3 *Denote*

$$q=\sum_j q_j,\quad Q=\sum_j Q_j,\quad R\ =\ q\mathbf{1}^T-Q. \tag{10}$$

If $\theta_1=\theta_2=\ldots=\theta$ *then the log-likelihood of the data* $\mathcal{S}_N$ *can be written as*

$$\ln P_{\theta,\sigma}(\mathcal{S}_N)\ =\ -\theta L_\sigma(R)-T\ln\psi(\theta). \tag{11}$$

Note that q_i, $Q_{ii'}$ represent respectively the number of times item i is observed in the data and the number of times item i' precedes i in the data.

Proposition 1 and Corollary 3 show that the infinite model $P_{\theta,\sigma}$ has *sufficient statistics*. The result is obtained without any assumptions on the lengths of the observed permutations. The data $\pi\in\mathcal{S}_N$ can have different lengths t_π, t_π may be unbounded, and may even be infinite.

As the parameters θ,σ of the model are infinite, the sufficient statistics R_j (or R) are infinite matrices. However, for any practically observed data set, t_{max} will be finite and the total number of items observed will be finite. Thus, N_j, R_j will be 0 for any $j > t_{max}$ and $R=\sum_{j\in\mathbb{P}}R_j$ will have non-zero entries only for the items observed in some $\pi\in\mathcal{S}_N$. Moreover, with a suitable relabeling of the observed items, one can reduce R_j to a matrix of dimension n, the number of distinct observed items. The rest of the rows and columns of R_j will be 0 and can be discarded. So in what follows we will assume that R_j and the other sufficient statistics have dimension n.

3.2 ML Estimation: The Case of a Single θ

We now go on to estimate θ and σ starting with the case of equal θ_j, that is, $\theta_1=\theta_2=\ldots=\theta$. In this case, Equation (11) shows that the estimation of θ and σ decouple. For any fixed σ, Equation (11) attains its maximum over θ at

$$\theta=\ln(1+T/L_\sigma(R)). \tag{12}$$

In contrast to the above explicit formula, for the finite GM, the likelihood has no analytic solution for θ (Fligner and Verducci, 1986). The estimated value of θ increases when $L_\sigma(R)$ decreases. This has an intuitive interpretation. The lower triangle of $\Sigma^T R \Sigma$ counts the "out of order" events w.r.t. the chosen model σ. Thus, $L_\sigma(R)$ can be seen as a residual, which is normalized by the "sample size" T. Equation (12) can be re-written as $L_\sigma(R)/T = 1/(e^\theta - 1) \equiv \xi$. In other words, the mean value parameter ξ equals the average residual. This recovers a well-known fact about exponential family models. In particular, if the residual is small, that is $L_\sigma(R)$ has low counts, we conclude that the distribution is concentrated, hence has a high θ.

Estimating σ^{ML} amounts to minimizing $L_\sigma(R)$ w.r.t σ, independently of the value of θ. The optimal σ according to Corollary 3 is the permutation that minimizes the lower triangular part of $\Sigma^T R \Sigma$. To find it we exploit an idea first introduced in Meilă et al. (2007). This idea is to search for $\sigma = (i_1|i_2|i_3|\ldots)$ in a stepwise fashion, starting from the top item i_1 and continuing down.

Assuming for a moment that $\sigma = (i_1|i_2|i_3|\ldots)$ is known, the cost to be minimized $L_\sigma(R)$ can be decomposed columnwise as

$$L_\sigma(R) \;=\; \sum_{l \neq i_1} R_{l i_1} + \sum_{l \neq i_1, i_2} R_{l i_2} + \sum_{l \neq i_1, i_2, i_3} R_{l i_3} + \ldots,$$

where the number of non-trivial terms is one less than the dimension of R. It is on this decomposition that the search algorithm is based. Reading the above algorithmically, we can compute $L_\sigma(R)$, for any given σ by the following steps

1. zero out the diagonal of R
2. sum over column i_1 of the resulting matrix
3. remove row and column i_1
4. repeat recursively from step 2 for $i_2, i_3, \ldots$.

If now σ is not known, the above steps 2, 3 become the components of a search algorithm, which works by trying every i_1 in turn, saving the partial sums, then continuing down for a promising i_1 value to try all i_2's that could follow it, etc. This type of search is represented by a *search tree*, whose nodes are candidate prefixes for σ.

The search tree has $n!$ nodes, one for each possible ordering of the observed items. Finding the lowest cost path through the tree is equivalent to minimizing $L_\sigma(R)$. *Branch-and-bound (BB)* (Pearl, 1984) algorithms are methods to explore the tree nodes in a way that guarantees that the optimum is found, even though the algorithm may not visit all the nodes in the tree. The number of nodes explored in the search for σ^{ML} depends on the individual sufficient statistics matrix R. It was shown by Meilă et al. (2007) that in the worst case, the number of nodes searched can be a significant fraction of $n!$ and as such intractable for all but small n. However, if the data distribution is concentrated around a mode, then the search becomes tractable. The more concentrated the data, the more effective the search.

We call the BB algorithm for estimating σ the SIGMA*, by analogy with the name A^* under which such algorithms are sometimes known. The algorithm is outlined in Figure 1. In this figure, A is an *admissible heuristic*; Pearl (1984) explains their role. By default, one can use $A \equiv 0$. A higher bound than 0 will accelerate the search; some of the admissible heuristics of Mandhani and Meilă (2009) can be used for this purpose.

In addition to this slow but exact algorithm, various heuristic search techniques can be used to explore the search tree of the problem. Two of them which showed good performance for the

standard GM model and which transfer immediately to the infinite model are the greedy search (GREEDYR) and the the SORTR heuristic of Fligner and Verducci (1988), both described in Figure 2. The SORTR computes the costs of the first step of BB, then outputs the permutation σ that sorts these costs in increasing order. The algorithm as proposed by Fligner and Verducci (1988) also performs limited search around this σ. For simplicity, this was not included in the pseudocode, but can be implemented easily.

The GREEDYR replaces BB with greedy search on the same sufficient statistics matrix. This algorithm was used by Cohen et al. (1999), where a factor of 2 approximation bound w.r.t. the cost was also shown to hold.

A third heuristic is related to the special case $t = 1$, when each π contains only 1 element. This is the situation of, for example, a search engine returning just the best match to a query. For $t = 1$, as Q is 0, the optimal ordering is the one minimizing $L_\sigma(q\mathbf{1}^T) = [0\ 1\ \ldots n-1]\Sigma^T q$. This is obviously the ordering that sorts the items in descending order of their frequency q.

In conclusion, to estimate the parameters from data in a single parameter case, one first computes the sufficient statistics, then a prefix of σ is estimated by exact or heuristic methods, and finally, with the obtained ordering of the observed items, one can compute the estimate of θ.

3.3 ML Estimation: The Case of General θ

Maximizing the likelihood of the data $\mathcal{S}_N$ is equivalent, by Proposition 1, with minimizing

$$J(\theta,\sigma) \;=\; \sum_j [\theta_j L_\sigma(R_j) + N_j \ln\psi(\theta_j)] \;=\; L_\sigma(\underbrace{\sum_j \theta_j R_j}_{R_\theta}) + \text{function of } \theta. \tag{13}$$

This estimation equation does not decouple w.r.t θ and σ. Minimization is however possible, due to the following two observations. First, for any fixed set of θ_j values, minimization w.r.t σ is possible by the algorithms described in the previous section. Second, for fixed σ, the optimal θ_j parameters can be found analytically by

$$\theta_j = \ln(1 + N_j / L_\sigma(R_j)). \tag{14}$$

The two observations immediately suggest an alternating minimization approach to obtaining θ^{ML}, σ^{ML}. The algorithm is given in Figure 3. For the optimization w.r.t σ exact minimization can be replaced with any algorithm that decreases the r.h.s of (13). As both steps increase the likelihood, the algorithm will stop in a finite number of steps at a local optimum.[4]

3.4 Identifiability and Consistency Results

One remarkable property of the IGM, which is easily noted by examining the likelihood in (8) or (11), is that the data will only constrain a finite number of parameters of the model. The log-likelihood (8) depends only on the parameters $\theta_{1:t_{max}}$. Maximizing likelihood will determine $\theta_{1:t_{max}}$ leaving the other θ_j parameters undetermined.

4. The reader may have noted that $P_{\theta,\sigma}$ is an exponential family model. For exponential family models with *continuous* parameters over a convex set, the likelihood is log-concave and an iteration like the one presented here would end at the global optimum. For our model, however, the parameter σ is discrete; moreover, the set of σ's forms the vertices of a convex polytope. One can show theoretically and practically that optimizing $L_\sigma(R)$ can have multiple optima and hence one cannot expect to always find a global maximum for the likelihood. However, we suspect that under the conditions that make optimization tractable, that is, a concentrated data distribution, existence of a global optimum may be proved.

Algorithm SIGMA*

Input matrix $R \in \mathbb{R}^{n\times n}$ of sufficient statistics

Initialize

$S = \{\bar{\sigma}_\emptyset\}$, $\bar{\sigma}_\emptyset$ =the empty ordering, $j = 0$, $C(\bar{\sigma}_\emptyset) = B(\bar{\sigma}_\emptyset) = 0$

Do

remove $\bar{\sigma} \in \underset{\bar{\sigma}\in S}{\operatorname{argmin}} B(\bar{\sigma})$ from S

if length$(\bar{\sigma}) = n$ *(Return)*

Output $\bar{\sigma}$, $B(\bar{\sigma}) = C(\bar{\sigma})$ and **Stop**.

else *(Expand* $\bar{\sigma}$*)*

for $i_{j+1} \in \{1:n\} \setminus \bar{\sigma}$

1. create node $\bar{\sigma}' = (i_1|\ldots,|i_j|i_{j+1})$
2. $v_{j+1}(\bar{\sigma}') = \sum_{l\in\{1:n\}\setminus\bar{\sigma}'} R_{li_{j+1}}$
3. calculate $C(\bar{\sigma}') = C(\bar{\sigma}) + v_{j+1}$, calculate $A(\bar{\sigma}')$
4. $B(\bar{\sigma}') = C(\bar{\sigma}') + A(\bar{\sigma}')$
5. store node $(\bar{\sigma}', j+1, C(\bar{\sigma}'), B(\bar{\sigma}'))$ in S

Figure 1: Algorithm SIGMA* outline. S is the set of nodes to be expanded; $\bar{\sigma} = (i_1|\ldots,|i_j)$ designates a top-j ordering, that is, a node in the tree at level j. The cost of the path $\bar{\sigma}$ is given by $C(\bar{\sigma}) = \sum_{j'=1}^{j}\sum_{l\notin\{i_{1:j'}\}} R_{li_l}$, and $A(\bar{\sigma})$ is a lower bound on the cost to go from $\bar{\sigma}$, possibly 0. The total estimated cost of node $\bar{\sigma}$ is $B(\bar{\sigma}) = C(\bar{\sigma}) + A(\bar{\sigma})$, which is used to predict which is the most promising path through the tree. In an implementation, node $\bar{\sigma}$ stores: $\bar{\sigma} = (i_1|\ldots,|i_j)$, $j = |\bar{\sigma}|$, $C(\bar{\sigma})$, $B(\bar{\sigma})$.

Let n be the number of distinct items observed in the data. From σ, we can estimate at most its restriction to the items observed, that is, the restriction of σ to the set $\bigcup_{\pi\in\mathcal{S}_N}\{\pi^{-1}(1),\pi^{-1}(2),\ldots,\pi^{-1}(t_\pi)\}$. The next proposition shows that the ML estimate will always be a permutation which puts the observed items before any unobserved items.

Proposition 4 *Let $\mathcal{S}_N$ be a sample of top-t orderings, and let σ be a permutation over $\mathbb{P}$ that ranks at least one unobserved item i_0 before an item observed in $\mathcal{S}_N$. Then there exists another permutation $\tilde{\sigma}$ which ranks all observed items before any unobserved items, so that for any parameter vector $(\theta_{1:t})$, $P_{\theta,\sigma}(\mathcal{S}_N) < P_{\theta,\tilde{\sigma}}(\mathcal{S}_N)$.*

Proof For an item i_0 not observed in the data, $q_{i_0,j}, Q_{i_0 i,j}$ and $R_{i_0 i,j}$ are 0, for any $j = 1:t$ and any observed item i. Hence row i_0 in any R_j is zero. Also note that if we switch among each other items that were not observed, there is no effect in the likelihood. Hence, w.l.o.g we assume that i_0 has rank j_0 in σ and is followed by an item i which is observed.

Algorithm SORTR

Input matrix $R \in \mathbb{R}^{n\times n}$ of sufficient statistics with 0 diagonal

1. compute the column sums of R, $r_l = \sum_k R_{kl}$, $l = 1:n$
2. sort r_l, $l = 1:n$ in increasing order

Output σ the sorting permutation

Algorithm GREEDYR

Input matrix $R \in \mathbb{R}^{n\times n}$ of sufficient statistics with 0 diagonal

1. set $V = 1:n$ the set of unused items
2. Repeat for $j = 1:n-1$
   (a) compute $r_l = \sum_{k\in V} R_{kl}$, $l \in V$ the column sums of a submatrix of R
   (b) let $l^* = \text{argmin}_{l\in V} r_l$
   (c) set $\sigma^{-1}(j) = l^*, V \leftarrow V \setminus \{l^*\}$
3. set $\sigma^{-1}(n)$ to the last remaining item in V

Output σ

Figure 2: Heuristic algorithms to estimate a central permutation: SORTR and GREEDYR. The elements R_{ii} are never part of any s_j, hence to simplify the code we assume they are set to 0.

Algorithm ESTIMATESIGMATHETA

Input Sufficient statistics R_j, N_j, $j = 1:t_{max}$
Initial parameter values $\theta_{1:t_{max}} > 0$

Iterate until convergence:

1. Calculate $R_\theta = \sum_j \theta_j R_j$
2. Find the ordering $\sigma = \text{argmin}_\sigma L_\sigma(R_\theta)$ (exactly by SIGMA* or by heuristics)
3. Estimate $\theta_j = \ln(1 + N_j / L_\sigma(R_j))$

Output σ, $\theta_{1:t_{max}}$

Figure 3: Algorithm ESTIMATESIGMATHETA.

The idea of the proof is to show that if we switch items i_0 and i the lower triangle of any R_j will not increase, and for at least one R_j it will strictly decrease.

For this, we examine row i of some R_j. We have $Q_{ii_0,j} = 0$ and $Q_{ii_0,j} = q_{i,j}$. Since i is observed then for at least one j we have $R_{ii_0,j} > 0$. Denote by σ' the permutation which is equal to σ except for switching i and i_0. The effect of switching i and i_0 on R_j is to switch elements $R_{ii_0,j}$ and $R_{i_0i,j}$. Since the latter is always 0 and the former is greater or equal to 0, it follows that $L_{\sigma'}(R_j) \leq L_\sigma(R_j)$ for any j, and that the inequality is strict for at least one j. By examining the likelihood expression in Equation (8), we can see that for any positive parameters $\theta_{1:j}$ we have $\ln P_{\theta,\sigma}(\mathcal{S}_N) < \ln P_{\theta,\sigma'}(\mathcal{S}_N)$.

By successive switches like the one described here, we can move all observed items before the unobserved items in a finite number of steps. Let the resulting permutation be $\tilde{\sigma}$. In this process, the likelihood will be strictly increasing at each step, therefore the likelihood of $\tilde{\sigma}$ will be higher than that of σ. □.

In other words σ^{ML} is a permutation of the observed items, followed by the unobserved items in any order. Hence the ordering of the unobserved items is completely non-identifiable (naturally so). But not even the restriction of σ to the observed items is always completely determined. This can be seen by the following example. Assume the data consists of the the two top-t orderings $(a|b|c)$, $(a|b|d)$. Then $(a|b|c|d)$ and $(a|b|d|c)$ are both ML estimates for σ; hence, it would be more accurate to say that the ML estimate of σ is the *partial ordering* $(a|b|\{c,d\})$. The reason σ^{ML} is not unique over a,b,c,d in this example is that the data has no information about the relative ranking c,d, neither directly by observing c,d together in the same π, nor indirectly, via a third item. This situation is likely to occur for the rarely observed items, situated near the ends of the observed π's. Thus this kind of inderterminacy will affect predominantly the last ranks of σ^{ML}. Another kind of indeterminacy can occur when the data is ambiguous w.r.t the ranking of two items c,d, that is, when $R_{cd} = R_{dc} > 0$. This situation can occur at any rank, and will occur more often for values of the θ_j parameters near 0. However, observing more data mitigates this problem. Also, since more observations typically increase the counts more for the first items in σ, this type of indeterminacy is also more likely to occur for the later ranks of σ^{ML}.

Thus, in general, there is a finite set of permutations of the observed items which have equal likelihood. We expect that these permutations will agree more on their first ranks and less in the last ranks. The exact ML estimation algorithm SIGMA* described here will return one of these permutations.

We now discuss the convergence of the parameter estimates to their true values. The IGM model has t real parameters θ_j, $j = 1:t$ and a discrete and infinite dimensional parameter, the central permutation σ. We give partial results on the consistency of the ML estimators, under the assumption that the true model is an IGM model and that t the length of the observed permutations is fixed or bounded.

Before we present the results, we need to make some changes in notation. In this section we will denote by $\hat{q}, \hat{R}$, etc the statistics obtained from a sample, normalized by the sample size N. We use q, R, etc for the asymptotic, population based expectation of a statistic under the true model $P_{\mathrm{id},\theta}$ (single parameter or multiple parameters as will be the case). For instance $\hat{q}_{i,j} = \sum_{\pi \in \mathcal{S}_N} q_{i,j}(\pi)/N$ represents the frequency of i appearing in position j in the sample, while $q_{i,j}$ is the probability of this event under $P_{\mathrm{id},\theta}$. For simplicity of notation, the dependence of N is omitted.

We will show that under weak conditions, the statistics of the type q, Q, R converge to their expectations, which in turn will entail convergence of the estimates based on them. The proofs are in Appendix A.1.

Proposition 5 *Let σ be any infinite permutation. If the true model is $P_{\mathrm{id},\theta}$ (multiple parameters) and t is fixed, then $\lim_{N\to\infty} L_\sigma(\hat{R}_j) = L_\sigma(R_j)$ for any $j = 1:t$.*

Since we can assume w.l.o.g. that the central permutation of the true model is the identity permutation, this proposition implies that for any IGM, with single or multiple parameters, and for any σ, the statistics $L_\sigma(\hat{R}_j)$ are consistent when t is constant.

Proposition 6 *If the true model is $P_{\mathrm{id},\theta}$ (multiple or single parameter) and t is fixed, then for any infinite permutation σ, denote by $\hat{\theta}_j(\sigma)$ (or $\hat{\theta}(\sigma)$) the ML estimate of θ_j (or θ) given that the estimate of the central permutation is σ. Then for $j = 1:t$*

$$\begin{aligned} \lim_{N\to\infty} \hat{\theta}_j(\sigma) &= \theta_j(\sigma), \\ \lim_{N\to\infty} \hat{\theta}(\sigma) &= \theta(\sigma), \end{aligned}$$

where the limits should be taken in the sense of convergence in probability.

Proposition 7 *Assume that the true model is $P_{\mathrm{id},\theta}$, t is fixed, and $\theta_j \geq \theta_{j+1}$ for $j = 1:t-1$. Let $\sigma \neq \mathrm{id}$ be an infinite permutation. Then, $P[L_{\mathrm{id}}(\hat{R}_j) < L_\sigma(\hat{R}_j)] \to 1$. Consequently, $P[L_{\mathrm{id}}(\hat{R}) < L_\sigma(\hat{R})] \to 1$.*

The consequences of these results are as follows. Assume the true model has a single parameter, and we are estimating a single parameter IGM model. Then, the likelihood of an infinite permutation σ is given by $\hat{R} = \sum_{j=1}^{t} \hat{R}_j$. By Proposition 7, for any σ other than the true one, the likelihood will be lower than the likelihood of the true permutation, except in a vanishingly small set of cases. This result is weaker than ideal, since ideally we would like to prove that the likelihood of the true σ is higher than that of all other permutations simultaneously. We intend to pursue this topic, but to leave the derivation of stronger and results for a further publication.

Proposition 6 shows that, if the correct σ is known, then the θ_j parameters, or alternatively the single θ parameter, are consistent.

In the multiple parameter IGM case, for any fixed θ, the likelihood of σ is given by $\hat{R} = \sum_{j=1}^{t} \theta_j \hat{R}_j$. Hence, by Proposition 7, in this case too, for any given σ different from the true central permutation, the likelihood of σ will be lower than the likelihood of the true model permutation.

The reader will note that these results can be easily extended to the case of bounded t.

4. Non-parametric Clustering

The above estimation algorithms can be thought of as finding a *consensus ordering* for the observed data. When the data have multiple modes, the natural extension to optimizing consensus is clustering, that is, finding the groups of the population that exhibit consensus.

Having defined a distance and a method for estimating ML parameters gives one access to a large number of the existing clustering paradigms originally defined for Euclidean data. For instance, the extensions of the K-means and EM algorithms to infinite orderings is immediate, and so are extensions to other distance-based clustering methods. Here we will present only one clustering method, a the *Exponential Blurring Mean-Shift (EBMS)*, but which illustrates well the issues of clustering in the space of top-t orderings. The EBMS is a nonparametric clustering method.

Algorithm EBMS

Input Top-t orderings $\mathcal{S}_N = \{\pi_i\}_{i=1:N}$, with length t_i; optionally, a scale parameter θ

1. For $\pi_i \in \mathcal{S}_N$ compute q_i, Q_i, R_i the sufficient statistics of a single data point.

2. Reduce data set by counting only the distinct permutations to obtain reduced $\tilde{\mathcal{S}}_N$ and counts $N_i \geq 1$ for each ordering $\pi_i \in \tilde{\mathcal{S}}_N$.

3. For $\pi_i, \pi_j \in \tilde{\mathcal{S}}_N$ calculate Kendall distance $d_{ij} = d_K(\pi_i, \pi_j)$.

4. (Optional, if θ not given in input) Set θ by solving the equation

$$E_\theta[d(\tilde{\mathcal{S}}_N)] = \frac{t_\pi e^{-\theta}}{1-e^{-\theta}} - \sum_{j=1}^{t_\pi} \frac{j e^{-\theta}}{1-e^{-j\theta}}$$

   where we set $E_\theta[d(\tilde{\mathcal{S}}_N)]$ to be the average of pairwise distances in step 3.

5. For $\pi_i \in \tilde{\mathcal{S}}_N$ *(Compute weights and shift)*

   (a) For $\pi_j \in \tilde{\mathcal{S}}_N$: set $\alpha_{ij} = \frac{\exp(-\theta d_{ij})}{\sum_{j'=1}^{n} \exp(-\theta d_{ij'})}$
   (b) Calculate $\bar{R}_i = \sum_{\pi_j \in \tilde{\mathcal{S}}_N} N_j \alpha_{ij} R_j$
   (c) Estimate σ_i the "central" permutation that optimizes $\bar{R}_i$ (exactly or by heuristics)
   (d) Set $\pi_i \leftarrow \sigma_i(1 : t_\pi)$

6. Go to step 2, until no π_i changes.

Output $\tilde{\mathcal{S}}_N$

Figure 4: The EBMS algorithm.

Nonparametric clustering is motivated by the fact that in many real applications the number of clusters is unknown and outliers exist. We consider an adapted version of the well known blurring mean-shift algorithm for ranked data (Fukunaga and Hostetler, 1975; Cheng, 1995; Carreira-Perpiñán, 2006). We choose the exponential kernel with bandwidth $\frac{1}{\theta} > 0$: $K_\theta(\pi, \sigma) = \frac{e^{-\theta d(\pi,\sigma)}}{\psi(\theta)}$. Under the Kendall distance $d_K(\pi, \sigma)$ the kernel has the same form as the one parameter Mallows' model. The kernel estimator of π_i is given by

$$\hat{r}(\pi_i) = \sum_{j=1}^{n} \frac{K_\theta(\pi_i, \pi_j)}{\sum_{k=1}^{n} K_\theta(\pi_i, \pi_k)} \pi_j = \sum_{j=1}^{n} \frac{e^{-\theta d_K(\pi_i, \pi_j)}}{\sum_{k=1}^{n} e^{-\theta d_K(\pi_i, \pi_k)}} \pi_j,$$

which does not depend on the normalizing constant $\psi(\theta)$ in Mallows' model.

The EBMS algorithm is summarized in Figure 4. It shift the "points" (i.e., top-t orderings) to new locations obtained by a locally weighted combination of all the data. Thus, every π is

"attracted" towards its closest neighbors; as the shifting is iterated the data collapse into one or more clusters. The algorithm has a *scale parameter* θ. The scale influences the size of the local neighborhood of a top-t ordering, and thereby controls the granularity of the final clustering: for small θ values (large neighborhoods), points will coalesce more and few large clusters will form; for large θ's the orderings will cluster into small clusters and singletons. In the EBMS algorithm, we estimate the scale parameter θ at each iteration by solving the equation in step (d).

Practical experience shows that blurring mean-shift merges the points into compact clusters in a few iterations and then these clusters do not change but simply approach each other until they eventually merge into a single point (Carreira-Perpiñán, 2006). Therefore, to obtain a meaningful clustering, a proper stopping criterion should be proposed in advance. For ranked data, this is not the case: since at each iteration step we round the local estimator into the nearest permutation, the algorithm will stop in a finite number of steps, when no ordering moves from its current position. Moreover, because ranked data is in a discrete set, we can also perform an accelerating process. As soon as two or more orderings become identical, we replace this *cluster* with a single ordering with a weight proportional to the cluster's number of members. The total number of iterations remains the same as for the original exponential blurring mean-shift but each iteration uses a data set with fewer elements and is thus faster.

In the algorithm one evaluates distances between top-t orderings. There are several ways in which to turn $d_K(\pi,\sigma)$ into a $d(\pi_1,\pi_2)$, where both terms are top-t orderings, containing different sets of items. Critchlow (1985) studied them, and here we adopt for $d(\pi_1,\pi_2)$ what is called the set distance, that is, the distance between the sets of infinite orderings compatible with π_1 respectively π_2.

We chose this formulation rather than others because this distance equals 0 when $\pi_1 = \pi_2$, and this is good, one could even argue necessary, for clustering. In addition, it can be calculated by a relatively simple formula, inspired by Critchlow (1985). Let

A	the set intersection of orderings π_1,π_2	t_1,t_2	lengths of π_1,π_2
B	$\pi_1 \setminus A$, the items in π_1 not in π_2	$n_{A,B,C}$	the number of items in A, B, C
C	$\pi_2 \setminus A$, the items in π_2 not in π_1	k_j	the index in π_1 of the j-th item not in A
		l_j	the index in π_2 of the j-th item not in A

Then

$$d(\pi_1,\pi_2) \;=\; d_K((\pi_1)_{|A},(\pi_2)_{|A}) + n_B n_C + n_B t_1 - \sum_{j=1}^{n_B} k_j - \frac{n_B(n_B-1)}{2} + n_C t_2 - \sum_{j=1}^{n_C} l_j - \frac{n_C(n_C-1)}{2}. \tag{15}$$

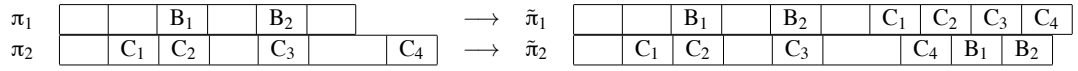


Figure 5: An example of obtaining two partial orderings $\tilde{\pi}_1,\tilde{\pi}_2$ compatible respectively with π_1,π_2 that achieve the set distance. Empty spaces represent the common items, while B and C symbols mark the items in B, respectively C. The distance $d(\pi_1,\pi_2)$ is the Kendall distance between $\tilde{\pi}_1$ and $\tilde{\pi}_2$.

The intuitive interpretation of this distance is given in Figure 5. We extend π_1 and π_2 to two longer orderings $\tilde{\pi}_1, \tilde{\pi}_2$, so that: (i) $\tilde{\pi}_1, \tilde{\pi}_2$ have identical sets of items, (ii) $\tilde{\pi}_1$ is the closest ordering to π_2 which is compatible with π_1, and (iii) reciprocally, $\tilde{\pi}_2$ is the closest ordering to π_1 which is compatible with π_2. We obtain $\tilde{\pi}_1$ by taking all items in π_2 but not in π_1 and appending them at the end of π_1 while preserving their relative order. A similar operation gives us $\tilde{\pi}_2$. Then, $d(\pi_1,\pi_2) = d_K(\tilde{\pi}_1,\tilde{\pi}_2)$, and Equation (15) expresses this value.

5. The Conjugate Prior

The existence of sufficient statistics implies the existence of a conjugate prior (DeGroot, 1975) for the parameters of model (6). Here we introduce the general form of this prior and show that computing with the conjugate prior (or posterior), is significantly harder than computing with the likelihood (6).

We shall assume for simplicity that all top-t rankings have the same t. Consequently, our parameter space consists of the real positive vector $\theta_{1:t}$ and the discrete infinite parameter Σ.

We define the prior parameters as a set of "fictitious sufficient statistics", by analogy with the sufficient statistics for model (6). For this we first make a few straightforward observations about the sufficient statistics q_j, Q_j, $j = 1:t$ as follows:

$$\begin{aligned} Q_1 &\equiv 0, \\ Q_{ii',j} &\geq 0 \quad \text{for all } i,i',j, \\ \sum_i q_{i,j} &= N_j \quad \text{for all } j, \\ Q_j\mathbf{1} &= (j-1)q_j \quad \text{for all } j>1. \end{aligned}$$

Therefore

$$R_j = q_j\mathbf{1}^T - Q_j = \begin{cases} Q_j\left(\frac{1}{j-1}\mathbf{1}\mathbf{1}^T - I\right) & \text{for } j>1 \\ q_1\mathbf{1}^T & \text{for } j=1 \end{cases}.$$

Now we let ν denote the *prior strength*, representing the equivalent sample size, and λ_1, Λ_j, $j = 2:t$ be the prior parameters corresponding to the sufficient statistics q_1, $Q_{2:t}$, normalized as follows.

Proposition 8 *Let $\nu > 0$, λ_1 be a vector and Λ_j, $j = 2:t$ denote a set of possibly infinite matrices satisfying*

$$\begin{aligned} \lambda_1 &\geq 0, \\ \Lambda_{ii',j} &\geq 0 \quad \text{for all } i,i',j, \\ \Lambda_j\mathbf{1} &= (j-1)\lambda_j \quad \text{for all } j>1 \quad \text{(by definition)}, \\ \mathbf{1}^T\lambda_j &= 1 \quad \text{for all } j. \end{aligned}$$

Denote $\mathbf{\Lambda} = \{\nu, \lambda_1, \Lambda_{2:t}\}$ and

$$R_j^0 = \begin{cases} \Lambda_j\left(\frac{1}{j-1}\mathbf{1}\mathbf{1}^T - I\right) & \text{for } j>1 \\ \lambda_1\mathbf{1}^T & \text{for } j=1 \end{cases}.$$

Define the distribution

$$P_{\mathbf{\Lambda}}(\sigma,\theta) \propto e^{-\nu\sum_{j=1}^t[\theta_j L(\Sigma^T R_j^0\Sigma)+\ln\psi(\theta_j)]}, \tag{16}$$

which is a conjugate prior for the model $P_{\theta,\sigma}(\pi)$ defined in (6).

Proof Given observed permutations $\pi_{1:N}$ with sufficient statistics R_j, $j = 1:t$, the posterior distribution of (σ, θ) is updated by

$$\begin{aligned} P(\theta, \sigma \,|\, \mathbf{\Lambda}, \pi_{1:N}) &\propto e^{-\sum_{j=1}^{t}[(\nu L_\sigma(R_j^0)+L_\sigma(R_j))\theta_j+(N+\nu)\ln\psi(\theta_j)]}, \\ &= e^{-(N+\nu)\sum_{j=1}^{t}[\theta_j L_\sigma\left(\frac{\nu R_j^0+R_j}{N+\nu}\right)+\ln\psi(\theta_j)]}. \end{aligned}$$

If the hyperparameters $\nu, \lambda_1, \Lambda_{2:t}$ satisfy the conditions of the proposition, then the new hyperparameters $\mathbf{\Lambda}' = \{\nu+N, (\nu\lambda_1+q_1)/(\nu+N), (\nu\Lambda_j+Q_j)/(\nu+N), j = 2:t\}$ satisfy the same conditions. □.

The conjugate prior is defined in (16) only up to a normalization constant.[5] As it will be shown below, this normalization constant is not always computable in closed form. Another aspect of conjugacy is that one prefers the conjugate hyperparameters to represent expectations of the sufficient statistics under some $P_{\theta,\sigma}$. The conditions in Proposition 8 are necessary, but not sufficient to ensure this fact.

To simplify the notations, we write

$$S_j^* \;=\; L_\sigma(\nu R_j^0 + R_j). \tag{17}$$

This notation reflects the fact that S_j^* is the counterpart in the posterior of the s_j in the distribution $P_{\theta,\sigma}(\pi)$. If $N = 0$, then $S_j^* = \nu L_\sigma(R_j^0)$. The value of S_j^* depends on σ and the hyperparameters, but does not depend on θ. The following result shows that for any fixed S_j^*, the posterior can be integrated over θ_j in closed form.

Proposition 9 *Let $P_{\mathbf{\Lambda}}(\sigma, \theta)$ be defined as in (16) and S_j^* be defined by (17). Then,*

$$P_{\mathbf{\Lambda}}(\theta_j \,|\, \sigma) \;=\; Beta_{S_j^*, \nu+1}(e^{-\theta_j}),$$

where $Beta_{\alpha,\beta}$ denotes the Beta distribution.

Proof sketch Replacing $\psi(\theta_j)$ with its value (5) yields

$$P_{\mathbf{\Lambda}}(\theta_j \,|\, \sigma) \;\propto\; e^{-S_j^*\theta_j}(1-e^{-\theta_j})^\nu,$$

from which the desired result follows by a change of variable. □

As a consequence, we have that

$$P_{\mathbf{\Lambda}}(\sigma) \;\propto\; \prod_{j=1}^{t} Beta(S_j^*(\sigma), 1+\nu). \tag{18}$$

In the above, the notation $Beta(x, y)$ is used to denote the special function Beta defined as $Beta(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$.

We have shown thus that closed form integration over the continuous parameters θ_j is possible. The summation over the discrete parameters poses much harder problems. We list them here.

5. The general form of a conjugate prior may include factors in σ and θ which do not depend on Λ. For simplicity of the exposition, we do not consider such a form here.

A first unsolved question is the range of the variables S^*_j. While the s_j variables in the infinite GM model are always integers ranging from 0 to infinity, the S^*_j variables can have non-integer values if ν or Λ_j are non-integer. The latter is almost always the case, since under the conditions of Proposition 8, Λ_j is not integer unless all its elements are 0 or 1. Second, Λ_j must have an infinite number of non-null entries, which may create problems for its numerical representation. And finally, there can be dependencies between S^*_j values for different j's. Hence, the factored expression (18) *should not be interpreted* as implying the independence of the S^*_j's.

We illustrate these points by a simple example. Assume that the conjugate prior hyperparameters are equivalent to the fictitious sample $\{\pi_1 = (1|2|3|\ldots), \pi_2 = (2|1|3|\ldots)\}$. Then,

$$\nu = 2,\ \lambda_1 = \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \end{bmatrix}\ \Lambda_2 = \begin{bmatrix} - & 0.5 & 0 \\ 0.5 & - & 0 \\ 0 & 0 & - \end{bmatrix},\ \Lambda_3 = \begin{bmatrix} - & 0 & 0 \\ 0 & - & 0 \\ 1 & 1 & - \end{bmatrix},$$

$$R^0_1 = \begin{bmatrix} - & 0.5 & 0.5 \\ 0.5 & - & 0.5 \\ 0 & 0 & - \end{bmatrix},\ R^0_2 = \begin{bmatrix} - & 0 & 0.5 \\ 0 & - & 0.5 \\ 0 & 0 & - \end{bmatrix},\ R^0_3 = \begin{bmatrix} - & 0 & 0 \\ 0 & - & 0 \\ 0 & 0 & - \end{bmatrix}.$$

For this example, there are two central rankings $\sigma_1 = (1|2|3|\ldots)$ and $\sigma_2 = (2|1|3|\ldots)$ which have the same $S^*_{1:3} = (1,0,0)$, but no σ with $S^*_j \equiv 0$. Assume now that we are given just $R^0_{1:3}$, $\nu = 2$ and $S^*_1(\sigma) = 1$ for some σ. Because $S^*_1 + 2$ is the sum of ranks of ${\pi^{-1}}_{1,2}(1)$ in σ, we can easily infer that the first two items in σ must be either $(1|2)$ or $(2|1)$ since any other σ will have $S^*_1 \neq 1$. But, for either of these possibilities, the computation of $S^*_2(\sigma)$ from R^0_2 gives $S^*_2(\sigma) = 0$. Hence, knowing S^*_1 informs about S^*_2 (in fact determines it completely), showing that S^*_1, S^*_2 are not independent.

Due to the above difficulties, computing the normalization constant of the posterior is an open problem. However, under some restrictive conditions, we are able to compute the normalization constant of the posterior in closed form.

Proposition 10 *If ν and $\Lambda_{1:t}$ are all integer, the S^*_j variables are independent, and the range of values of S^*_j is $\mathbb{P}$ then*

$$P(S^*_j = k) \ = \ (N+\nu)Beta(k+1, N+1+\nu),$$

and consequently

$$P_{\Lambda,\nu}(\theta_{1:t}, S^*_{1:t}) \ = \ (N+\nu)^t e^{-\sum_{j=1}^t [\theta_j S^*_j + (N+\nu)\ln\psi(\theta_j)]}.$$

The proof is given in Appendix A.1.

Now we examine the case of a single parameter IGM. The conjugate prior is given by $\nu > 0$ and a single matrix R^0 corresponding to the normalized sufficient statistics matrix R,

$$P_{\nu,R^0}(\theta,\sigma) \ \propto \ e^{-\theta L(\Sigma^T \nu R^0 \Sigma) + t\nu \ln\psi(\theta)}, \tag{19}$$

The posterior is

$$P_{\nu,R^0}(\theta,\sigma\,|\,\pi_{1:N}) \ \propto \ e^{-L(\theta\Sigma^T(\nu R^0 + R)\Sigma) + t(\nu+N)\ln\psi(\theta)]}.$$

Denote for simplicity $N' = N+\nu$, $R' = (R+\nu R^0)/(N+\nu)$, $S^*(\sigma) = L_\sigma(N'R')$. Then, the parameter θ again follows a Beta distribution given σ,

$$P_{N',R'}(\theta\,|\,\sigma) \ \propto \ Beta_{S^*;tN'+1}(e^{-\theta}).$$

After integrating θ out, we obtain

$$P_{N',R'}(\sigma) \;\propto\; Beta(S^*(\sigma), tN'+1).$$

Finally, let us note that that the priors in (16) and (19) are both informative with respect to σ. By replacing the term $L_\sigma(R_j^0)$ (respectively $L_\sigma(R^0)$) with some $r_j > 0$ (respectively $r > 0$) one can obtain a prior that is independent of σ, hence uninformative. However, this prior is improper.

6. Experiments

In this section, we conduct experiments on single θ estimation, general θ estimation, real data sets, and clustering.

6.1 Estimation Experiments, Single θ

In these experiments we generated data from an infinite GM model with constant $\theta_j = \ln 2, \ln 4$ and estimated the central permutation and the parameter θ. To illustrate the influence of t, t_π was constant over each data set. The results are summarized in Table 1.

Note that while θ appears to converge, the distance $d_K(\sigma^{ML}, \sigma)$ remains approximately the same. This is due to the fact that, as either N or t increase, n, the number of items to be ranked, increases. Thus the distance d_K will be computed between ever longer permutations. The least frequent items will have less support from the data and will be those misranked. We have confirmed this by computing the distance between the true σ and our estimate, restricted to the first t ranks. This was always 0, with the exception of $n = 200, \theta = 0.69, t = 2$ when it averaged 0.04 (2 cases in 50 runs) (A more detailed analysis of the ordering errors will be presented in the next subsection.)

Even so the table shows that most ordering errors are no larger than 1. We also note that the sufficient statistic R is an unbiased estimate of the expected R. Hence, for any fixed length $\tilde{t}$ of σ^{ML}, the σ estimated from R should converge to the true σ (see also Fligner and Verducci, 1988). The θ^{ML} based on the true σ is also unbiased and asymptotically normal.

6.2 Estimation Experiments, General θ

We now generated data from an Infinite GM model with $\theta_1 = \ln 2$ or $\ln 4$ and $\theta_j = 2^{-(j-1)/2}\theta_1$ for $j > 1$. As before, t_π was fixed in each experiment at the values 2, 4, 8. We first look at the results for $t = 8$ in more detail. As the estimation algorithm has local optima, we initialized the θ parameters multiple times. The initial values were (i) the constant value 0.1 (chosen to be smaller than the correct values of all θ_j), (ii) the constant values 1 and respectively 2 depending whether $\theta_1 = \ln 2$ or $\theta_1 = \ln 4$ and (iii) the true θ parameters. The case (ii) ensured that the initial point is higher than all correct values for all the estimated θ_j.

Figure 6 shows the estimated values of θ_j for different sample sizes N ranging in $\{200, 500, 1000, 2000\}$. By comparing the respective (i) and (ii) panels, one sees that the final result was insensitive to the initial values and always close to the true θ_j. The results were also identical to the results for the initialization (iii), and this was true for all the experiments we performed. Therefore, in the subsequent plots, we only display results for one initialization, (i).

Qualitatively, the results are similar to those for single θ, with the main difference stemming from the fact that, with decreasing θ_j values, the sampling distribution of the data is spread more, especially w.r.t the lower ranks.

Estimates of θ (mean stdev)									
θ	**N**	200		500		1000		2000	
		mean	std	mean	std	mean	std	mean	std
0.69	$t=2$	0.68	0.04	0.68	0.03	0.68	0.03	0.68	0.024
	$t=4$	0.67	0.03	0.69	0.02	0.69	0.01	0.69	0.01
	$t=8$	0.68	0.02	0.69	0.01	0.69	0.01	0.69	0.007
1.38	$t=2$	1.34	0.13	1.37	0.09	1.39	0.05	1.37	0.04
	$t=4$	1.40	0.06	1.38	0.05	1.39	0.03	1.38	0.03
	$t=8$	1.37	0.03	1.38	0.03	1.38	0.02	1.38	0.01

Ordering error									
θ	**N**	200		500		1000		2000	
		$d_K=0$	$d_K=1$	$d_K=0$	$d_K=1$	$d_K=0$	$d_K=1$	$d_K=0$	$d_K=1$
0.69	$t=2$	0.42	0.36	0.28		0.36		0.40	0.38
	$t=4$	0.36	0.36	0.40		.44		0.32	0.30
	$t=8$	0.32	0.34	0.44		0.40		0.38	0.32
1.38	$t=2$	0.82	0.18	0.92	0.08	0.76	0.24	0.90	0.08
	$t=4$	0.92	0.08	0.92	0.08	0.88	0.12	0.88	0.10
	$t=8$	0.84	0.16	0.92	0.08	0.76	0.20	0.16	0.88

Number observed items n									
θ	**N**	200		500		1000		2000	
		mean	std	mean	std	mean	std	mean	std
0.69	$t=2$	9.24	0.92	10.76	0.66	11.88	1.16	12.68	1.07
	$t=4$	11.92	0.81	12.92	1.04	14.32	1.10	14.88	1.01
	$t=8$	16.04	0.89	17.36	0.99	18.16	1.07	19.40	0.91
1.38	$t=2$	5.68	0.74	6.04	0.79	6.76	0.78	7.32	0.55
	$t=4$	7.72	0.84	8.16	0.74	8.68	0.75	9.48	0.82
	$t=8$	11.52	0.65	12.52	0.58	13.24	0.66	13.40	0.71

Table 1: Results of estimation experiments, single parameter IGM. Top: mean and standard deviation of θ^{ML} for two values of the true θ and for different t values and sample sizes N. Middle: the proportion of cases when the ordering error, that is, the number inversions w.r.t the true σ^{-1} was 0, respectively 1. Bottom: number of observed items n (mean and standard deviation). Each estimation was replicated 25 or more times.

This figure allows us to observe the "asymmetry" of the error in θ^{ML}. The estimates seem to biased towards larger values, especially for higher j and less data. There is a theoretical reason for this. Recall that by Equation (12) θ is a decreasing function of $L_\sigma(R)$. If the true σ is not optimal for the given R, due to sample variance, then θ^{ML} will tend to overestimate θ. Hence θ^{ML} is a *biased* estimate of θ. If however, due to imperfect optimization, the estimated σ^{ML} is not optimal and has higher cost than σ, then θ^{ML} will err towards underestimation. In Figures 6 and 7 the bias is always positive, indicating that the minimization over σ is done well (even though it is not guaranteed to reach optimality).

Now we consider the recovery of the central permutation σ. From our previous remarks, we expect to see more ordering errors in the bottom ranks of σ^{ML}, where the distribution is less concentrated (smaller θ_j) and there is less data available. To visualize these effects easier, it is interesting to look at rankings with small t. When t is small, 2, 4 or 8, the total number of items to be ranked (Figure 8) is several times larger than t for our experiments. Thus the estimation algorithm has to put together an ordering over this many items, when only groups of $t = 2, 4, \ldots$ were observed together. As an additional confounding effect, the top elements will be oversampled, so the information about the lower ranks will have to be inferred indirectly by pooling the information from the whole data set. This is what the algorithm is doing, and Figure 9 shows how well it succeeds in that.

The figure displays the ordering error between σ^{ML} and the true σ for each rank, which is measured by $s_j(\sigma^{ML}|\sigma)$. Recall that the total number of inversions between σ^{ML} and σ is the sum of all s_j; similarly, the total number of errors in σ^{ML} up to rank r is given by $\sum_{j=1}^{r} s_j(\sigma^{ML}|\sigma)$. All plots illustrate the same general tendency of the s_j values to increase *slowly* with j. The increase is slower when there are more observations per rank, that is when N and t are larger, and when the θ_j are larger (thus the data distribution is more concentrated).

6.3 Experiments on Real Data Sets, With General θ, Tied Parameters

The next experiment was conducted with the data collected by Cohen et al. (1999). The data consists of a list of 157 universities, the queries, and a set of 21 search engines, the "experts". Each search engine outputs a list of up to $t_{max} = 30$ URL's when queried with the name of the university. The data set provides also a "target" output for each query, which is the university's home page.

Hence, we have 147 estimation problems (10 universities with no data), with sample size $N \leq 21$ (as some experts return empty lists) and with variable length data ranging from $t = 1$ to $t = 30$. Figure 10 gives a summary view of number of samples for each rank N_j, $j = 1 : t$, the number of distinct items n and the cumulative number of ranks observed (i.e., $T = \sum_{j \leq 30} N_j$). These values suggest that estimating a fully parameterized model with distinct $\theta_{1:30}$ may lead to overfitting and therefore we estimate several parameterizations, all having the form $\Theta_r = (\theta_1, \theta_2, \ldots \theta_{r-1}, \theta_r, \theta_r, \ldots \theta_r)$. In other words, ranks $1 : r-1$ have distinct parameters, while the following ranks share parameter θ_r. We call $\theta_{1:r-1}$ the *free* parameters and θ_r the *tied* parameter. For $r = 1$ we have the single parameter model, and for $r = t_{max} = 30$ we have the fully parameterized model.

Estimating a model with r parameters is done by a simple modification of the ESTIMATESIGMATHETA algorithm which is left to the reader.

The estimation algorithm was started from the fixed value $\theta_j = 0.1$ for all runs. The number of iterations to convergence range between 10 and 50, with typical value 18. The running time was around per model estimated.

In Figure 11 we give a synopsis of the values of the θ parameters under different models. The single parameter models yields θ values in the range [0.007, 0.104] with the 10%, 50% and 90% quantiles being respectively 0.009, 0.018,and 0.032. The parameters θ are on average decreasing in all models, with the free parameters higher than the tied parameters for the remaining ranks. This is true on average only, while for individual samples some of the free parameters may be smaller than the tied parameter.

Notice also that for the models with fewer parameters the values of the free parameters tend to be higher than the corresponding values in models with more parameters. Compare for instance the values of θ_1 in the two-parameter model with θ_1 in the 30 parameter model.

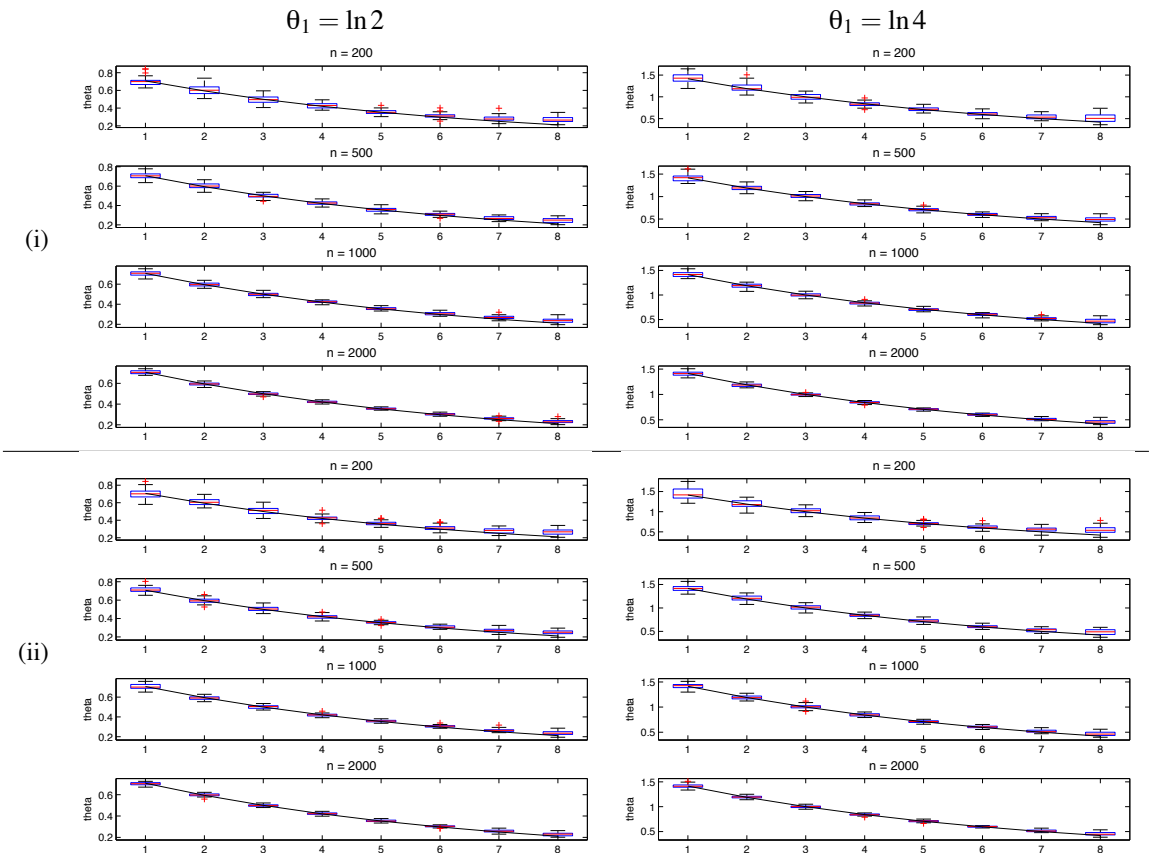


Figure 6: Estimation of the parameters $\theta_{1:t}$ for $t = 8$, different sample sizes $N = 200, 500, 1000, 2000$, different true parameters θ, and different initializations: (i) $\theta_j \leftarrow 0.1$, (ii) $\theta_j \leftarrow 1$ or 2. For each experimental condition, the corresponding graph displays box plots of the obtained estimates of θ_j, $j = 1:8$ for 50 random samples, with the j on the horizontal axis. The continuous line crossing the box plots marks the true values of the parameters $\theta_{1:8}$ (exponential decay starting from the given θ_1).

For each query and each model size, we computed the rank of the true university home page, that is, the *target*, under the estimated central permutation σ^{ML}. Assuming the search engines are reasonably good, this rank is an indirect indicator of the goodness of a model. In addition, for each query, we selected one model by BIC and calculated the target ranks for these models. Table 2 gives the mean and median of the target rank for each model, as well as for the BIC selection. The rank is assigned to $t_{max} + 1 = 31$ if the target is not among the items returned by the search engines.

It is evident that while BIC's performance is better than selecting a one-parameter model, it is not optimal w.r.t the ranking of the target home page.

We used a modified form of the BIC criterion, that takes into account that the continuous parameters are not all estimated from the same sample size. We have derived[6] the following formula

6. The derivation is omitted, being outside the scope of this paper.

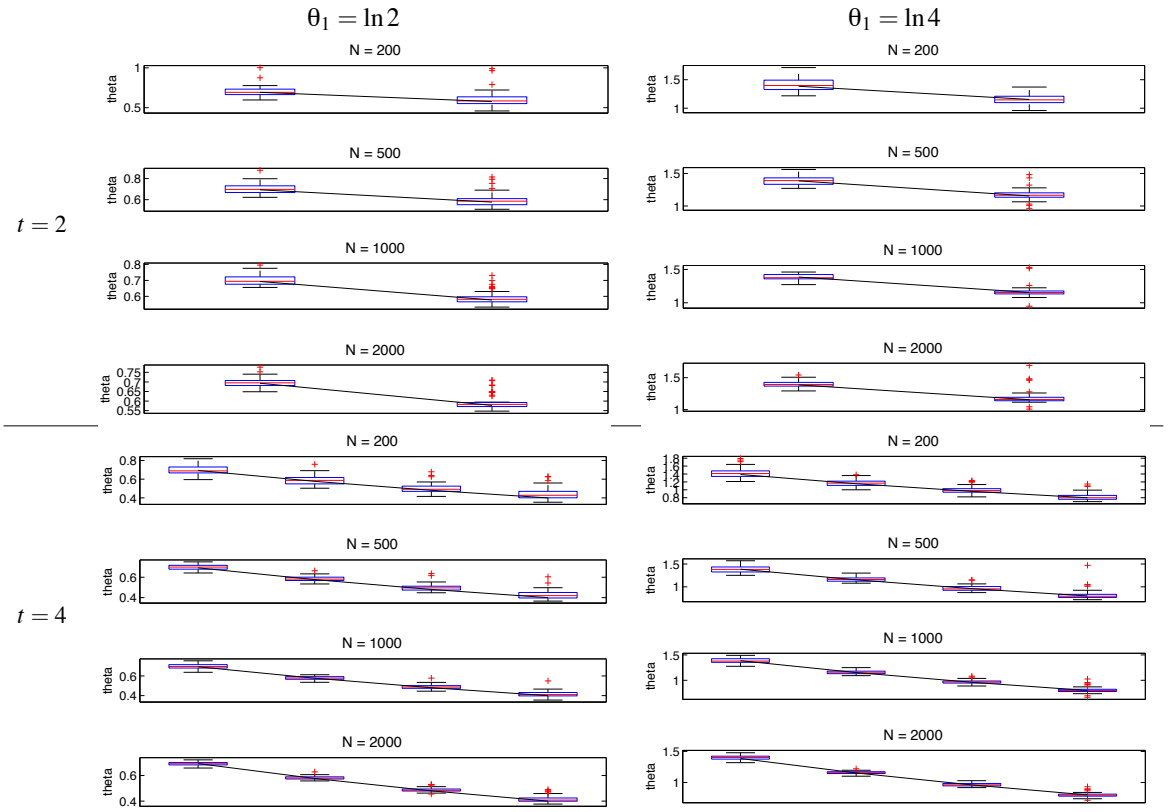


Figure 7: Same as Figure 6 for $t = 2$ and $t = 4$ and a single initial point $\theta_j \equiv 0.1$.

for BIC:

$$BIC(r) \;=\; \ln P_{\sigma,\theta}(\mathcal{S}_N) - \frac{1}{2}\sum_{j=1}^{r} \ln N'_j \tag{20}$$

where $N'_j = N_j$ for $j < r$ and $N'_r = \sum_{j'=r}^{t} N_j$. This expression approximates the marginal distribution of the model w.r.t the continuous parameter. The discrete parameter σ is not marginalized out. This parameter has always the same dimension, dictated by the observed data, and independent of r. In any finite data situation, the parameter will be finite. Therefore we can see the model selection problem as a model selection over r and a very large but finite set of discrete σ's. Maximizing the BIC in (20) is equivalent with maximizing the BIC over this much larger set of models, if one assumes that the ML estimation procedure attains a global optimum.

Next, we tested the ESTIMATESIGMATHETA algorithm on the Jester data of Goldberg et al. (2001).[7] This data set represents a set of 100 jokes, which were scored by approximately 25,000 people. From the numerical scores, we obtained a partial ordering over the jokes rated by each individual. Mao and Lebanon (2008) also analyzed this data set and found that it was multimodal. To obtain data sets closer to unimodality, we picked a person at random (this is person 945 in the data) and extracted the N nearest neighbors of this ranking, for $N = 200$ and $N = 12,000$. The smaller data set was expected to be more concentrated than the larger data set.

7. Available at `http://goldberg.berkeley.edu`.

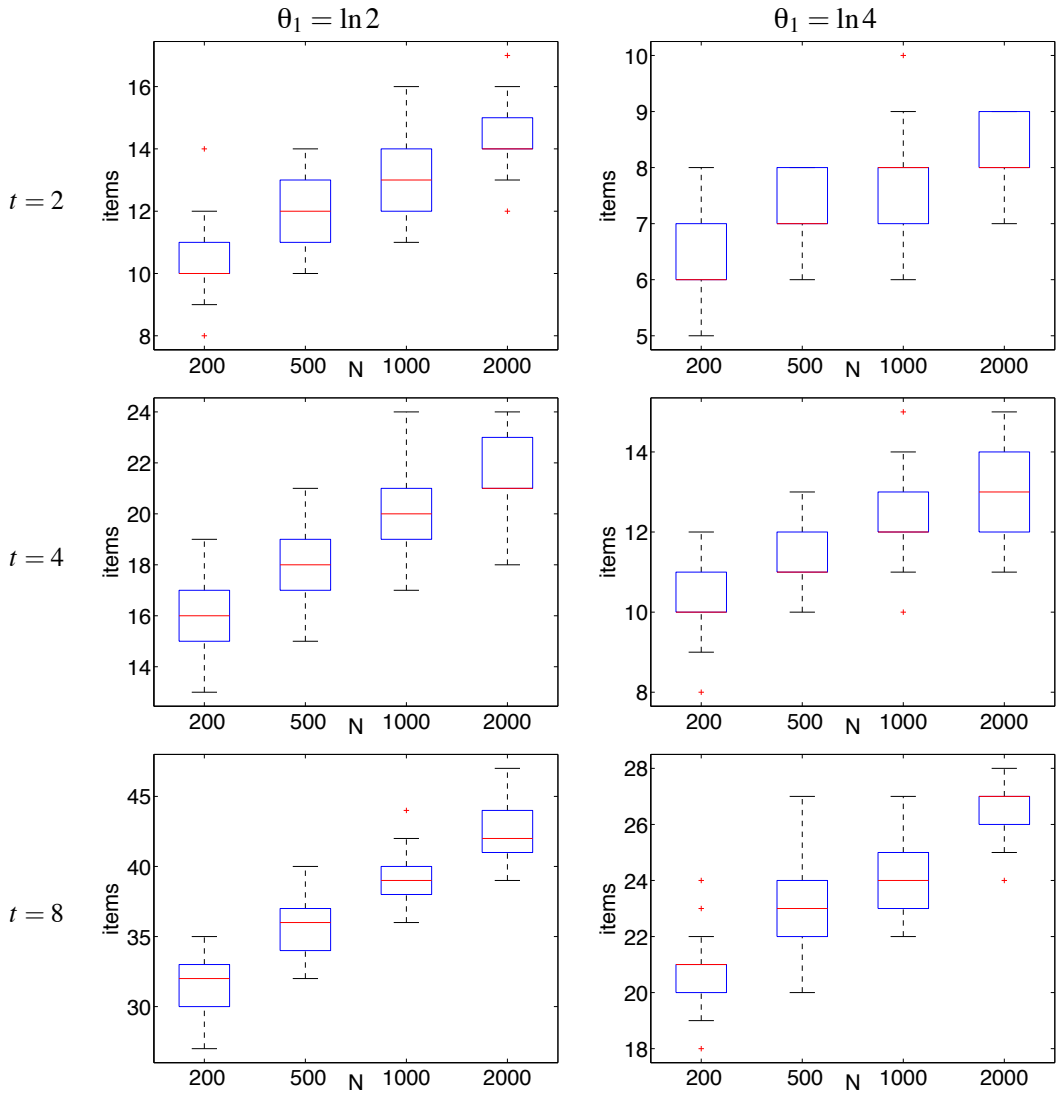


Figure 8: Number of items observed for $N = 200, 500, 1000, 2000$, different parameters θ and $t = 2, 4, 8$. Each box plot represents the distribution of the number of items over 50 random samples.

We ran the ESTIMATESIGMATHETA on this data set with different values of r. The log-likelihoods obtained on the training set and the BIC values are shown in Figure 12.

As expected, the likelihood is highest or nearly so for the model with maximum number of free parameters (for $N = 12,000$ the likelihood is not monotonic due to imperfect optimization over σ). However, the BIC is not monotonic. For the large N case, where the data is dispersed, it chooses a model with $r = 76$ parameters. The estimated θ_j values range in $[0.05, 0.07]$ for $j = 1 : 30$ (nearly all ranks that have $N_j = 12000$) but become higher, up to 0.2 for the ranks that have smaller N_j's. For the smaller and more concentrated data set, BIC has equal values for three different models: the

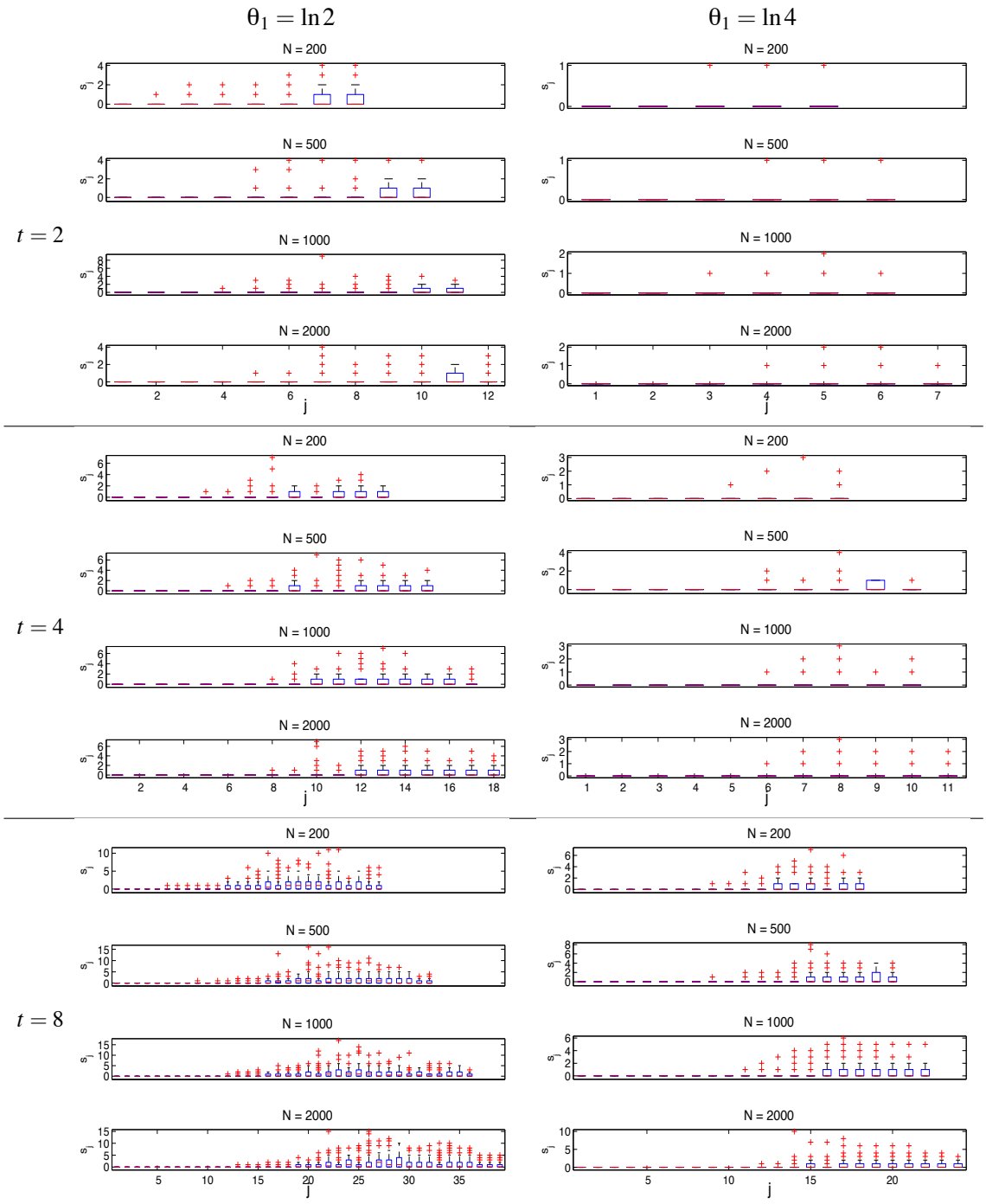


Figure 9: Ordering errors for $N = 200, 500, 1000, 2000$, different parameters θ_j , and $t = 2, 4, 8$. The error for a rank j is given by $s_j(\sigma^{ML}|\sigma^{true})$. Each box plot represents the distribution of s_j over 50 random samples.

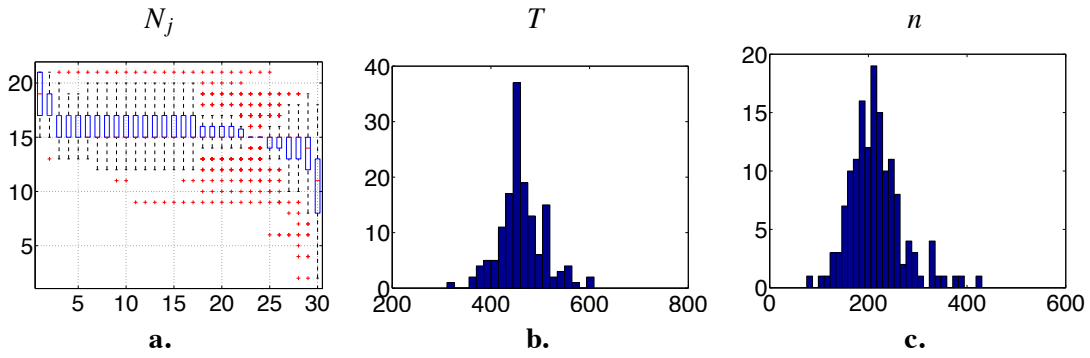


Figure 10: Summaries of the universities data: boxplots of the number of samples per rank, (a), histogram of total items observed T (b), histogram of the number n of distinct items observed (c), over all 147 queries.

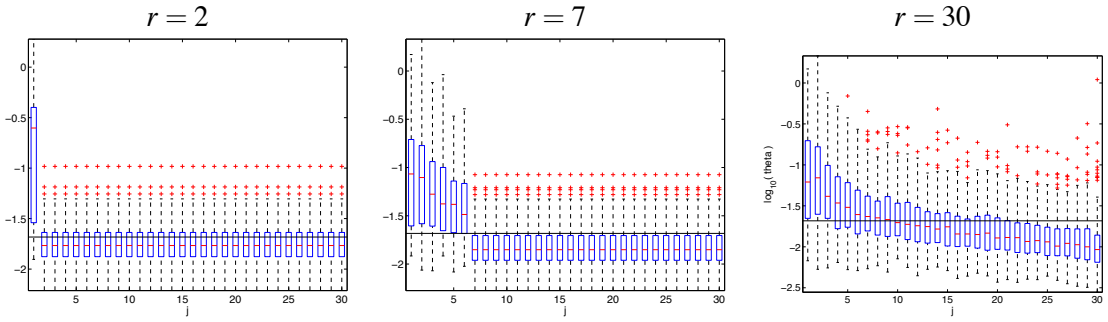


Figure 11: Boxplots of the Θ estimates over all queries for model with 2, 7, and 30 parameters. The vertical axis of the scale is *logarithmic, base 10*, that is, 0 corresponds to $\theta_j = 1$ and -2 to $\theta_j = 0.01$. For clarity, the distribution of the tied parameter (which is always the last parameter) is replicated for $j = r : t_{max}$. The horizontal line marks the mean value of θ in the single parameter model.

Model size	1	2	3	4	5	6	7	8	9	10	30	BIC
Mean rank (good)	5.3	5.7	4.2	4.2	**4.1**	**4.1**	4.4	4.5	5.0	5.2	5.1	5.5
Median rank (good)	3	3	**1.5**	2	2	2	2	2	2	3	3	3
Mean rank (all)	16.5	16.1	**15.4**	15.5	15.5	15.6	15.8	15.7	15.9	16.0	16.0	17.5
Median rank (all)	13	15	11	11	12	**9**	10	10	11	11	11	18

Table 2: Mean and median of the rank of the target web page under each model, and under the BIC selected model. These statistics are computed once over all 147 universities and once over a subset of 41 universities where the target is always ranked in the first 30; the subset is labeled as "good".

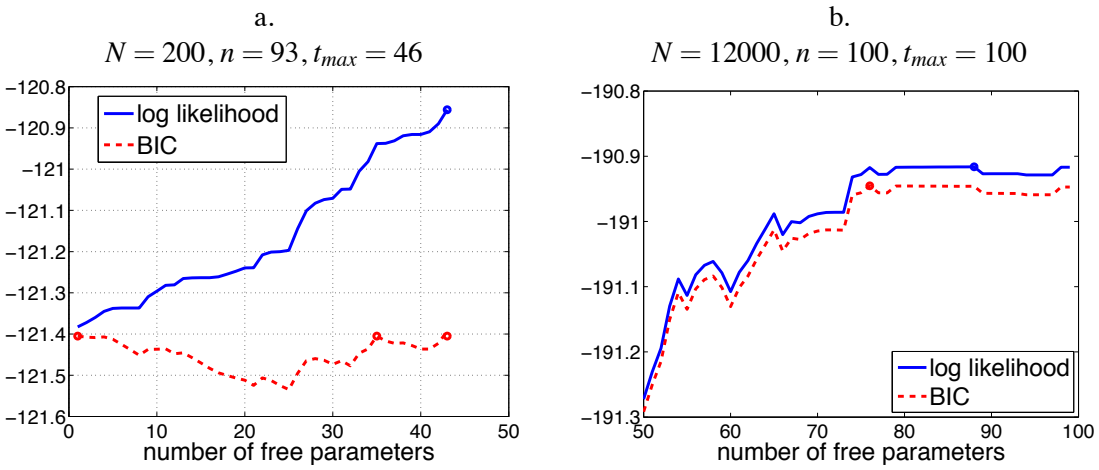


Figure 12: Log likelihood and BIC values per data point on two subsamples from the Jester data set. The circles mark the maxima of the log-likelihood, respectively BIC.

single parameter model, the full parameter model, and an intermediate model with $r = 36$. This is not so suprising as it may appear, since $N_j = N$ for $j \geq 36$, and the estimated θ_j values in this range are all very close to 0.1. Thus, the three models will effectively differ only in the way they model ranks $37 : 43$.

This finding suggests that the single parameter (infinite) Mallows model is a good model locally, in the space of the `jester` data. The large N experiment indicates otherwise, which agrees both with our common sense assumption that this data is multimodal, and with the finding of Mao and Lebanon (2008). In Mao and Lebanon (2008) a non-parametric Mallows model was used (more about this model in Section 7.3); our experiment supports their use of a single parameter model.

6.4 Clustering Experiments

The first experiment was with artificial data. We generated sample orderings with 3 clusters of 150 rankings each. Each cluster k was sampled from an Infinite GM model with a single spread parameters θ_k, with θ_1, θ_2, θ_3 equal to 1.5, 1.0, 0.7 respectively. The cluster centers are random permutations of infinitely many objects. In addition, each data set contains 50 outliers. In each data set all data had the same length t. We experimented with $t_\pi = 4, 6, 8$.

We ran the Exponential Blurring Mean-Shift, K-means, and EM Model-based clustering algorithms 10 times on samples from this distribution. For EBMS, the scale parameter was estimated based on the average of pairwise distances. In step 5c of the algorithm, the new ranking can be much longer than the original partial ranking. As seen above, the last ranks are subject to noise and overfitting. Therefore we truncated the new ranking to the length of number of observed items.

For the K-means and model-based algorithms, we experiment with different numbers of clusters, and report the best classification error with respect to the true clustering. This puts these two

algorithms at an advantage w.r.t EBMS, but as Table 3 shows, even so the nonparametric algorithm achieves the best performance.

Note that the error rate in Table 3 is computed including the outliers, that is, we compared a true clustering with 53 clusters (3 clusters and 50 singletons) to the clustering obtained when the algorithm converged.

For EM and K-means the number of clusters associated with the lowest classification errors was between 3 and 5. From the table we see that the K-means and model based approach identified three primary clusters correctly. K-means did not have the ability of identifying the outliers, so it just assigned each outlier into one of those primary clusters. The model-based approach assigned outliers into primary clusters too, but it also gave more uncertainty on the outliers (the probabilities of outliers belonging to their assigned cluster were relatively smaller than data from primary clusters).

The running time per data set of EBMS was under a minute, and the number of iterations to convergence followed the pattern typical of mean-shift algorithms and was never larger than 10.

Next, we examine data on college course preferences in the Republic of Ireland. Each prospective student applies by ranking up to 10 degree course in order of preference. Extensive details of the college applications system are available at `http://www.cao.ie`.

The data used in our analysis was previously studied by Gormley and Murphy (2006). They found that the geographical positions of the institution had a significant influence on choice of courses which complicated the interpretation of the vocational callings. Our analysis focuses on the subset of students who applied to Trinity College Dublin (TR) and University College Dublin (DN), both located in the capital of Ireland. These two universities offered $n = 228$ degree courses. There were 1095 female and 862 male applicants who only put TR and DN courses in their top-5 preferences. The EBMS algorithm is applied to top-5 rankings for the female and male applicants separately.

Table 4 shows these clustering results. For the female applicants the first cluster mostly consists of Art, Law and Business courses. Since the largest cluster contains about three quarters of the data, we run the EBMS clustering again for the applicants within this cluster and find four major subgroups in term of vocational callings: Law, Business, Drama, and English. The clustering results for male applicants show 5 clusters, plus a singleton. We run EBMS again for the largest cluster, and find three major sub-groups in term of vocational callings: Finance, Law, and History. As Gormley and Murphy (2006) discovered in their experiments, for the subset of Dublin applicants, there are differences between the clustering of females and males in the central permutations, but similarities too. The main similarity is that the grouping is vocational. Each group contains courses in both universities, with no strong preference for one versus the other. Second, a large proportion of both genders opt for business, economics and law disciplines. For the females, the Arts courses are also highly favored. As Gormley and Murphy (2006) explains, the Arts course is a broad liberal arts

Top-t rankings	EBMS	K-means	EM
$t = 4$	0.0030 (0.0001)	0.1014 (0.0038)	0.1008 (0.0025)
$t = 6$	0.0014 (0.0001)	0.0986 (0.0010)	0.1000 (0.0000)
$t = 8$	0.0002 (0.0001)	0.0972 (0.0010)	0.1000 (0.0000)

Table 3: Classification Errors: mean and standard deviation of 10 random samples.

course which can be followed with many different specializations later on. Hence, its high ranking in several clusters is an indication that female candidates want to leave more options open later. In general, the central rankings of each cluster are very clearly separating the pool of candidates into various profiles.

Males

cluster size	657	143	35	13	13
1st choice	BESS (TR)	Engineering (TR)	Science (DN)	Science (DN)	Medicine (TR)
2nd choice	Commerce (DN)	Science (DN)	Science (TR)	Science (TR)	Medicine (DN)
3rd choice	Business and Law (DN)	Engineering (DN)	Mathematics (TR)	Medicine (TR)	Pharmacy (TR)
4th choice	Arts (DN)	Computer Science (DN)	Theoretical Physics (TR)	Medicine (DN)	Dental Science (TR)
5th choice	Economics and Finance (DN)	Computer Science (TR)	Theoretical Physics (DN)	Pharmacy (TR)	Veterinary Medicine (DN)

Females

cluster size	725	162	141	41	26
1st choice	Arts (DN)	Arts (DN)	Arts (DN)	Physiotherapy (DN)	Physiotherapy (TR)
2nd choice	Law (DN)	Psychology (DN)	Psychology (DN)	Physiotherapy (TR)	Physiotherapy (DN)
3rd choice	BESS (TR)	Psychology (TR)	Psychology (TR)	Radiation Therapy (TR)	Science (DN)
4th choice	Business and Law (DN)	Science (DN)	Law (DN)	Radiography (DN)	Medicine (DN)
5th choice	Law (TR)	Science (TR)	Social Science (DN)	Occupational Therapy (TR)	Medicine (TR)

Table 4: EBMS Clustering of female and male applicants. BESS stands for Business, Economic and Social Science, TR for Trinity College Dublin and DN for University College Dublin.

7. Discussion and Related Work

In this section, we discuss the relation between IGM and GM, other related models and algorithms, and draw the brief conclusion.

7.1 Relation to the GM Model

It is useful to compare the various aspects of the IGM presented here with the respective aspects of the standard GM. We do so now, highlighting also which of them were already published and which are new.

- s **representation.** This was introduced by Fligner and Verducci (1986) for the GM model. For finite number of items n, j ranges in $1:n-1$ and s_j in $0:n-j+1$.

- **marginal distributions, $P_{\theta,\sigma}$, over top-t orderings.** It is also introduced by Fligner and Verducci (1986). The main difference with the IGM model is in the normalization constant, which has the expression

$$\prod_{j=1}^{t}\psi_j(\theta_j) \quad \text{with} \quad \psi_j(\theta_j) = \frac{1-e^{-(n-j+1)\theta_j}}{1-e^{-\theta_j}}. \tag{21}$$

  Also, for the GM model, the underlying space $\mathcal{S}_n$ is finite, while for the IGM, $\mathcal{S}_{\mathbb{P}}$ is uncountable.

- **sufficient statistics as in Proposition 1.** Proposition 1 represents a new result for the GM as well. The only difference is in the replacement of $\psi(\theta_j)$ with $\psi_j(\theta_j)$ from (21) in (8). The

nearest previous result is that of Meilă et al. (2007) which establishes sufficient statistics for the GM model over complete permutations.

If we have a single parameter GM model and complete permutations, then it is easy to see that $\sum_j Q_j$ represents the sufficient statistics for σ alone. In computer science estimating σ in this context is called the *consensus ranking* problem, or the *minimum feedback arc set* problem. In this case, by setting $t = n$ for all permutations, the sufficient statistics defined in (8) reduce to the previously known $\sum_j Q_j$. Thus, our main contribution in this respect is to prove that not knowing θ, and not observing complete permutations still results in an exponential family model with sufficient statistics.

- θ **estimation.** In the GM case, this is a convex unidimensional optimization solved numerically (Fligner and Verducci, 1986; Meilă et al., 2007).

- σ **estimation by** SIGMA* The SIGMA* and ESTIMATESIGMATHETA algorithms can be used for the GM model as well, with the estimation of θ performed numerically. The closed form Equation (12) can serve as a very good initial point for the iterative optimization algorithm. Note that while the heuristics SORTR and GREEDYR are simple, they could not have been applied before to the GM model over top-t orderings because it was not known that this model has sufficient statistics for σ.

- **conjugate prior** Fligner and Verducci (1990) introduced an informative prior for θ, which had a single "sufficient statistics" parameter. They used it with a uniform prior over σ, noting that this prior cannot be normalized or integrated analytically. The informative conjugate prior for θ and σ introduced in Section 5 applies also to the standard GM. Again, the main formal change is replacing $\psi(\theta_j)$ by $\psi_j(\theta_j)$. With this change, we lose the elegant closed form integration over θ proved in Proposition 9. The GM conjugate prior will not be in general integrable in closed form over θ. The uninformative prior for the IGM becomes of course a proper prior in the GM case. If we set all the r_j parameters to the same value, we obtain exactly the same prior as Fligner and Verducci (1990).

  Often a non-informative prior is used as a regularizer for the ML estimator, turning it into a *Maximum A-Posteriori (MAP)* estimator. This is possible for the IGM and GM too. All one needs to do is to replace, in the inputs to the estimation algorithms, the data sufficient statistics with the posterior sufficient statistics.

- EBMS **clustering** The algorithm adapts seamlessly to finite number of items.

7.2 Other Models and Algorithms for Finite Permutations

This work acknowledges its roots in the work of Fligner and Verducci (1986) on stagewise ordering models and in the recent paper Meilă et al. (2007). The latter shows for the first time that GM models have sufficient statistics, and describes an exact but non-polynomial algorithm to find the central permutation. While similarities exist between the algorithm of Meilă et al. (2007) and the SIGMA* algorithm presented here, we stress that our representation (based on the inversion table $(s_j)_{j=1:t}$) is *different* from the representation (denoted V_j) in Meilă et al. (2007).

In fact, the two representations could be called reciprocal, as for any given complete permutation π, finite or infinite, $s_j(\pi|\mathrm{id}) = V_j(\pi^{-1}|\mathrm{id})$. This difference is trivial if complete permutations are

observed, but not for missing data. In particular, the distribution of V_j for top-t orderings does not seem to have sufficient statistics for $j > 2$ even in the case of finite permutations. The s_j representation has another advantage that V_j has not: for any finite data set, a parameter s_j is either completely determined or completely undetermined the data, whereas in the reciprocal V_j representation *all* V_j are weakly constrained by data.

While both our SIGMA* and the algorithm of Meilă et al. (2007) perform branch-and-bound search on a matrix of sufficient statistics, the sufficient statistics in this paper are derived by an entirely different method, and cannot be obtained by naively replacing the sufficient statistics of Meilă et al. (2007).

It may be noted that the cost $L_\sigma(R)$ bears a striking similarity to the cost function used by Quadrianto et al. (2010) in the context of matching by the method of kernelized sorting. The latter cost function can be expressed as $\operatorname{trace} K\Sigma^T L\Sigma$, to be minimized over Σ. The authors show that this problem is quadratic in the matrix Σ when K, L are symmetric, positive definite matrices and use a quadratic relaxation algorithm to optimize it efficiently. The cost $L_\sigma(R)$ can be rewritten as $\operatorname{trace} Q_0\Sigma^T R\Sigma$ where Q_0 is the upper triangular matrix defined in the proof of Proposition 2. The main difference between our cost function and the one of Quadrianto et al. (2010) is the fact that ours involves the non-symmetric matrices Q_0, R and is not a quadratic problem in Σ.

An interesting application of the GM model to multimodal data is Lebanon and Lafferty (2003), where the σ's play the role of the data, so the parameter estimation is done entirely differently. In an early work Critchlow (1985) examines several classes of (Haussdorf) distances for partial orderings. Murphy and Martin (2003) cluster ranking data by the EM algorithm and in Gormley and Murphy (2005, 2006) the EM is used for the purpose of analyzing Irish voting patterns and college applications. The base model used by the latter papers is is not the Mallows model but the Plackett-Luce model (Plackett, 1975; Luce, 1959). The estimation of this model from data is much more difficult and, as Gormley and Murphy (2005) show, can be only done approximately. Busse et al. (2007) use the s_j representation in the context of EM clustering of partial orderings, without however recognizing the existence of sufficient statistics.

A greedy algorithm for consensus ordering with partially observed data is introduced in Cohen et al. (1999). Meilă et al. (2007) show that their cost function optimized is closely related to the log-likelihood of the Mallows' model, using a modified form of the Q matrix defined in (10). This algorithm, like GREEDYR, does not estimate a θ parameter. Cohen et al. (1999) introduce a computational improvement based on interpreting a value $Q_{ii'} > 0.5$ as an arc from i to i'. They note that it is sufficient to search for the optimal permutation in each *strongly connected component* of the resulting directed graph, which can sometimes greatly reduce the dimension of the search space.

If a permutation π is not complete, Cohen et al. (1999) replaces the unobserved $Q_{ii'}(\pi)$ with the value 0.5. This ad-hoc procedure allows the GREEDYR to run on top-t rankings, but it is not statistically correct, since the optimized cost will not be a likelihood. If we use the correct matrix of sufficient statistics R, then the reduction procedure based on strongly connected components does not apply any more.

7.3 Other Models and Algorithms For Infinite Permutations

All the above works deal with permutations on finite sets. In fairness to Cohen et al. (1999) we remark that their work, although non-rigorous with respect to incomplete permutations, is motivated

by the same problem as ours, that is, dealing with a very large set of items, of which only some are ranked by the "voters".

The paper of Thoma (1964) studies the space of infinite permutations which differ from the identity in a finite number of positions. In the vocabulary of the present paper, these would be the infinite permutations at finite distance d_K from σ. In a single parameter infinite GM, these infinite permutations are the only ones which have non-zero probability. While from a probabilistic perspective the two views are equivalent, from a practical perspective they are not. We prefer to consider in our sample space all possibile orderings, including those with vanishing probability. It is the latter who are more representative of real experiments. For instance, in the university web sites ranking experiment, our model assumed that there is a "true" central permutation from which the observations were generated as random perturbations. This is already an idealization. But we also have the liberty to assume that the observations are very long orderings which are close to the central permutation only in their highest ranks, and which can diverge arbitrarily far from it in the latter ranks. We consider this a more faithful scenario than assuming in addition that the observation must be identical to the central permutation (and hence to each other!) on all but a finite number of ranks.

Recently Mao and Lebanon (2008) introduced a kernel density estimator and estimation algorithm, that elegantly allows partial orderings of a large variety of *types* to be modeled together. The kernel is the single parameter Mallows' model, with θ as kernel width. One of the interesting contributions of this paper is an algorithm for averaging the $d(\tilde{\pi}_1, \tilde{\pi}_2)$ over all (infinite) permutations $\tilde{\pi}_1, \tilde{\pi}_2$ compatible with given partial orderings π_1, π_2. The relation with EBMS is evident. It is also evident that within EBMS one could incorporate the average distance as calculated by Mao and Lebanon (2008) instead of the current set distance, with everything else staying the same. Since the average distance is always larger than the set (minimum) distance for top-t permutations, and in particular it is not 0, the optimal kernel width θ will have different values.

7.4 Conclusion

We have introduced a natural extension of stagewise ordering to the the case of infinitely many items. The new probabilistic model preserves the elegant properties of its finite counterpart: it has sufficient statistics, an exact estimation algorithm (albeit intractable in the worst case) and tractable heuristics that work well when the data come from a unimodal distribution. Sampling, distance computations, clustering extend to this class of models in a natural way.

We have paid particular attention to non-parametric clustering by mean-shift blurring, showing by experiments that the algorithm is practical and effective. This illustrates our view of the utility of the IGM model. The IGM, an exponential model with a simple, intuitive distribution, should be seen as a building block for more complex distributions, as needed by the data at hand. For instance, the extension to finite mixtures (that is, parametric clustering and multimodal distributions) is immediate. It is also an open problem to extend the kernel density estimator of Mao and Lebanon (2008) to infinite models and GM models with multiple parameters.

We are not aware of any statistical work on estimating parametric models over infinite orderings. There are also no previous results on sufficient statistics for finite partial orderings, so the present paper can be said be first in this respect as well.

One important advantage of having a model with multiple parameters, with n finite or infinite, is that each rank can be modeled by a separate parameter θ_j (the standard GM/IGM) or one can

tie the parameters of different ranks (the way we did in Section 6). This way, one can use larger θ_j values to penalize the errors in the first ranks more, and smaller θ_j values for the lower ranks, where presumably more noise in the orderings is permissible. This property of the model fits well with human perception of "distance" between orderings, or with the noise we may expect in human generated data.

Tying the parameters for the lower ranks is also important for reducing variance. If the observed data have different lengths t, then necessarily $N_j \geq N_{j'}$ for $j' > j$. In other words, for larger j's we may have less data available to estimate θ_j. Tying the parameters has the benefic effect of smoothing the θ_j values, as it was shown in all the experiments with real data. Another way to smooth the parameters is to use an uninformative prior as a regularizer. This way, too, each θ_j can be regularized separately by the hyperparameter r_j. This hyperparameter has a clear meaning—it is the expectation of s_j in the fictitious sample; therefore, a user can easily tune the strength of the prior using Equation (7) with r_j in place of ξ_j. This equation will give for any r_j a value θ_j^0 towards which the θ_j value will be shrunk. Shrinkage via the conjugate prior can be done for the finite GM as well, except that the relation (7) will be implicit instead of closed form.

Beyond its mathematical elegance and simplicity of use, we believe that an infinite model has practical importance as well. In many instances the number of items to rank is very large. Search engines come immediately to mind, understood as algorithms for retrieval by inexact matching from a large database. Under this umbrella fall not only the well-known web search engines, but also the various specialized algorithms for finding matches in biological data bases, like Sequest (Eng et al., 1994) and Blast (Altschul et al., 1990). These algorithms output ranked lists, from which the human user interprets only the top t entries. The data base, that is, the set of items, is usually not fixed; typically it is growing as more proteins, genes, web pages are discovered. It is natural under this scenario to assume that n is potentially infinite. As we have shown, this does not make working with the data more difficult, and occasionally makes it faster.

Acknowledgments

Thanks to Jon Wellner for bringing up the question of infinite permutations. This work was partially supported by NSF awards IIS-0313339 and IIS-0535100. A preliminary version of the IGM model was published as Meilă and Bao (2008).

Appendix A. Proofs

We give the proofs for the asymptotic results in the following subsection.

A.1 Proofs for the Asymptotic Results

Proof of Proposition 5 We start with the observation that under $P_{\mathrm{id},\theta}$ any observed top-t ranking π is a function of the variables $s_{1:t}$ who are independently distributed according to discrete exponential laws. For each s_j the empirical CDF converges to the true CDF and, as we know, this entails the fact that for any function f over $\mathbb{N}$, the sample expectation of f converges to the true expectation of f; see for example, van der Vaart (1998). This also holds for the joint distribution of $s_{1:t}$ and functions of $s_{1:t}$. In other words, the sample expectation of any function $f(\pi)$ is consistent, when π ranges over all top-t permutations.

Now consider $L_\sigma(R_j)$ for a fixed σ. By definition, $L_\sigma(R_j(\pi)) = s_j(\pi|\sigma)$ and $L_\sigma(\hat{R}_j)$ is the sample expectation of $s_j(\pi|\sigma)$. According to the definition of $s_j(\pi|\sigma)$ in (2), this is a function of $\pi^{-1}(1),\ldots\pi^{-1}(j)$ and the fixed σ. It follows by the argument above that $L_\sigma(\hat{R}_j)$ converges to $L_\sigma(R_j)$ for any j and any σ.

Note that for $\sigma = \mathrm{id}$ the argument is simpler, since $L(R_j) = s_j$. □

We now refer to a property of the Mallows model introduced by Fligner and Verducci (1988). An IGM $P_{\sigma,\theta}$ has *complete consensus* if for any two items i, i' with $i \prec_\sigma i$, we have that $P[i \prec_\pi i'] > P[i' \prec_\pi i]$. The following result is a modified form of Theorem 2 of Fligner and Verducci (1988) that applies to truncated infinite permutations.

Proposition 11 (Complete consensus) *The IGM model $P_{\sigma,\theta}$ with*

$$\theta_j \geq \theta_{j+1}, \tag{22}$$

has complete consensus. Moreover, condition (22) entails that for any fixed t, any $j = 1:t$, and any items i, i' with $\sigma(i) < \sigma(i')$

$$R_{ii',j} \;<\; R_{i'i,j}.$$

Proof Fix i, i' as above. Let π be a (complete) permutation where $i \prec_\pi i'$. Let us denote by π' the permutation obtained by transposing i and i' in π; denote $\pi(i) = k$, $\pi(i') = k'$, $k' > k$.

We want to show that under the condition of the proposition, $P_{\sigma,\theta}(\pi) \geq P_{\sigma,\theta}(\pi')$. We first observe that

$$s_j(\pi') = \begin{cases} s_j(\pi) & \text{for} \quad j < k \,\text{or}\, j > k' \\ s_j(\pi) + i' - i - r & \text{for} \quad j = k \,\text{and}\, r = |\{x \,|\, i < \sigma(x) < i', \pi(x) < k\}| \\ s_j(\pi) \,\text{or}\, s_j(\pi) + 1 & \text{for} \quad k < j < k' \\ s_j(\pi) - r' & \text{for} \quad j = k' \,\text{and}\, r' = |\{x \,|\, i < \sigma(x) < i', \pi(x) > k'\}| \end{cases}.$$

Note also that $r + r' \leq i' - i - 1$ or, in other words, $i' - i - r > r'$. Now, we look at the likelihood ratio $P_{\sigma,\theta}(\pi)/P_{\sigma,\theta}(\pi')$:

$$\begin{aligned} \ln[P_{\sigma,\theta}(\pi)/P_{\sigma,\theta}(\pi')] &= \sum_{j=k}^{k'} \theta_j [s_j(\pi') - s_j(\pi)], \\ &\geq \theta_k [s_k(\pi') - s_k(\pi)] + \theta_{k'} [s_{k'}(\pi') - s_{k'}(\pi)], \\ &= \theta_k (i' - i - r) - \theta_{k'} r', \\ &\geq \theta_k r' - \theta_{k'} r' \;=\; (\theta_k - \theta_{k'}) r' \;\geq\; 0. \end{aligned}$$

It follows that $P_{\sigma,\theta}(\pi) \geq P_{\sigma,\theta}(\pi')$. Moreover, if $\pi(i) = j, \pi(i') = j+1$ for some j then this inequality is strict. Now let $A = \{\pi \,|\, i \prec_\pi i'\}$ be the set of all permutations π as defined above; its complement B equals $\{\pi' \,|\, i' \prec_{\pi'} i\}$. It is immediate that from the above that $P_{\sigma,\theta}(A) = P_{\sigma,\theta}(i \prec i') > P_{\sigma,\theta}(B) = P_{\sigma,\theta}(i' \prec i)$, which proves the first claim of the proposition.

For the second claim, fix i, i' with $\sigma(i) < \sigma(i')$, a rank j and another rank $j' > j$. Take π such that $\pi(i) = j$ and $\pi(i') = j'$, and let π' be the permutation obtained by transposing i and i' in π. Then obviously $P_{\sigma,\theta}(\pi) > P_{\sigma,\theta}(\pi')$. $R_{ii',j}$, by definition, is the total probability of permutations of the form π with $j' = j+1, j+2, \ldots$, while $R_{i'i,j}$ is the total probability of permutations of the form π'. Therefore, $R_{ii',j} > R_{i'i,j}$. □

Proof of Proposition 5 From Proposition 11 it follows that if the true central permutation is the identity, then $L_{\mathrm{id}}(R_j) < L_\sigma(R_j)$ for any $\sigma \neq \mathrm{id}$. Let $\varepsilon = L_\sigma(R_j) - L_{\mathrm{id}}(R_j) > 0$. Then, because of the consistency of $L_\sigma(R_j)$ for any σ, it follows that $P_{\mathrm{id},\theta}[L_{\mathrm{id}}(\hat{R}_j) - L_\sigma(\hat{R}_j) > L_{\mathrm{id}}(R_j) - L_\sigma(R_j) + \varepsilon = 0] \to 0$. □

Proof of Proposition 6 This proof follows from the consistency of $L_\sigma(R_j)$. Since the ML estimate of θ_j is a continuous function of $L_\sigma(R_j)$ it will be consistent as well. Similarly, for the single parameter IGM, the ML estimate of θ is a continuous function of $(L_\sigma(R_j),\, j = 1:t)$, and therefore it is consistent as well. □

Proof of Proposition 10 Given the observed rankings $\{\pi_{1:N}\}$ and the hyperparameters ν, $\Lambda_{1:t}$ the marginal distribution of the central permutation σ can be expressed in terms of S_j^*:

$$P(S_j^* = k) \;\propto Beta(k+1, N+1+\nu) \;=\; \frac{\Gamma(k+1)\Gamma(N+1+\nu)}{\Gamma(k+N+2+\nu)} \;=\; f(k),$$

$$\frac{f(k+1)}{f(k)} \;=\; \frac{\Gamma(k+2)\Gamma(N+1+\nu)}{\Gamma(k+N+3+\nu)} \div \frac{\Gamma(k+1)\Gamma(N+1+\nu)}{\Gamma(k+N+2+\nu)} \;=\; \frac{k+1}{N+k+2+\nu},$$

$$f(k) \;=\; \frac{k!}{(N+2+\nu)\cdots(N+1+\nu+k)} \times f(0) \;=\; \frac{k!}{(N+1+\nu)\cdots(N+1+\nu+k)}.$$

We prove in Lemma 12 that $\sum_{k=0}^{\infty} f(k) = \frac{1}{(N+\nu)}$. Therefore, the normalization constant of $P(S_j^*)$ is $1/(N+\nu)$ and the conclusion of the Proposition follows. □

Lemma 12 $\sum_{k=0}^{\infty} \frac{k!}{(N+\nu+1)\cdots(N+\nu+1+k)} = \frac{1}{(N+\nu)}$.

Proof of Lemma 12 We write the general term of the series as a difference

$$\begin{aligned}
\frac{1}{N+\nu} - \frac{1}{(N+\nu+1)} &= \frac{1!}{(N+\nu)(N+\nu+1)}, \\
\frac{1!}{(N+\nu)(N+\nu+1)} - \frac{1!}{(N+\nu+1)(N+\nu+2)} &= \frac{2!}{(N+\nu)(N+\nu+1)(N+\nu+2)}, \\
&\vdots
\end{aligned}$$

Through mathematical induction we can prove that

$$\sum_{k=0}^{K} \frac{k!}{(N+\nu+1)\cdots(N+\nu+1+k)} + \frac{(K+1)!}{(N+\nu)(N+\nu+1)\cdots(N+K+\nu+1)} = \frac{1}{(N+\nu)}.$$

Moreover, for fixed N, ν,

$$\begin{aligned}
\frac{(K+1)!}{(N+\nu)(N+\nu+1)\cdots(N+K+\nu+1)} &= \frac{(K+1)!(N+\nu-1)!}{(N+K+\nu+1)!}, \\
&= \frac{(N+\nu-1)!}{(K+2)(K+3)\cdots(N+K+\nu+1)}, \\
&= O(k^{-(N+\nu)}).
\end{aligned}$$

From this the desired result follows. □

References

S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 3(213):403–410, 1990. doi:10.1006/jmbi.1990.9999., PMID 2231712.

L.M. Busse, P. Orbanz, and J. Bühmann. Cluster analysis of heterogeneous rank data. In *Proceedings of the International Conference on Machine Learning ICML*, 2007.

M.A. Carreira-Perpiñán. Fast nonparametric clustering with gaussian blurring mean-shift. In *23rd International Conference on Machine Learning (ICML)*, pages 153–160, 2006.

Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17 (8):790–799, 1995. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.400568.

W.C. Cohen, R.S. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

D.E. Critchlow. *Metric methods for analyzing partially ranked data*. Number 34 in Lecture notes in statistics. Springer-Verlag, Berlin Heidelberg New York Tokyo, 1985.

M.H. DeGroot. *Probability and Statistics*. Addison–Wesley Pub. Co., Reading, MA, 1975.

J.K. Eng, A.L. McCormack, and J.R. Yates. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society of Mass Spectrometry*, 5:976–989, 1994. doi:10.1016/1044-0305(94)80016-2.

M.A. Fligner and J.S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society B*, 48:359–369, 1986.

M.A. Fligner and J.S. Verducci. Multistage ranking models. *Journal of the American Statistical Association*, 88:892–901, 1988.

M.A. Fligner and J.S. Verducci. Posterior probability for a consensus ordering. *Psychometrika*, 55: 53–63, 1990.

K. Fukunaga and L.D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Information Theory*, IT-21:32–40, 1975. ISSN 0018-9448.

K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, July 2001. http://goldberg.berkeley.edu/jester-data/, to cite for Jester data set.

I.C. Gormley and T.B. Murphy. Analysis of irish third-level college applications. *Journal of the Royal Statistical Society, Series A*, 169(2):361–380, 2006.

I.C. Gormley and T.B. Murphy. Exploring heterogeneity in irish voting data: A mixture modelling approach. Technical Report 05/09, Department of Statistics, Trinity College Dublin, 2005.

G. Lebanon and J. Lafferty. Conditional models on the ranking poset. In *Advances in Neural Information Processing Systems*, number 15, Cambridge, MA, 2003. MIT Press.

R.D. Luce. *Individual Choice Behavior*. Wiley, New York, 1959.

C.L. Mallows. Non-null ranking models. *Biometrika*, 44:114–130, 1957.

B. Mandhani and M. Meilă. Better search for learning exponential models of rankings. In David VanDick and Max Welling, editors, *Artificial Intelligence and Statistics AISTATS*, number 12, 2009.

Y. Mao and G. Lebanon. Non-parametric modelling of partially ranked data. *Journal of Machine Learning Research*, 9:2401–2429, 2008. URL `jmlr.csail.mit.edu/papers/v9/lebanon08a.html`.

M. Meilă and L. Bao. Estimation and clustering with infinite rankings. In David McAllester and Petri Millimäki, editors, *Proceedings of the 24-th Conference on Uncertainty in Artificial Intelligence (UAI 2008)*. AUAI Press, 2008.

M. Meilă, K. Phadnis, A. Patterson, and J. Bilmes. Consensus ranking under the exponential model. In Ron Parr and Linda Van den Gaag, editors, *Proceedings of the 23rd Conference on Uncertainty in AI*, volume 23, page (to appear), 2007.

T.B. Murphy and D. Martin. Mixtures of distance-based models for ranking data. *Computational Statistics and Data Analysis*, 41(3–4):645–655, 2003.

J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

R.L. Plackett. The analysis of permutations. *Applied Statistics*, 24:193–202, 1975.

N. Quadrianto, A.J. Smola, L. Song, and T. Tuytelaars. Kernelized sorting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (preprint), 2010.

R.P. Stanley. *Enumerative Combinatorics*, volume 1. Cambridge Unversity Press, Cambridge, New York, Melbourne, 1997.

E. Thoma. Die unzerlegbaren, positiv-definiten Klassenfunctionen der abzälbar unendlichen, symmetrische Gruppen. *Mathematische Zeitschrift*, 85:40–61, 1964.

A.W. van der Vaart. *Asymptotic statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambrigde University Press, 1998.

Journal of Machine Learning Research 11 (2010) 3519-3540 Submitted 10/09; Revised 8/10; Published 12/10

Rate Minimaxity of the Lasso and Dantzig Selector for the ℓ_q Loss in ℓ_r Balls

Fei Ye FYE@DRW.COM
Department of Quantitative Research
DRW Trading Group
Chicago, IL 60661-2555, USA

Cun-Hui Zhang CZHANG@STAT.RUTGERS.EDU
Department of Statistics and Biostatistics
Rutgers University
Piscataway, NJ 08854-8019, USA

Editor: Hui Zou

Abstract

We consider the estimation of regression coefficients in a high-dimensional linear model. For regression coefficients in ℓ_r balls, we provide lower bounds for the minimax ℓ_q risk and minimax quantiles of the ℓ_q loss for all design matrices. Under an ℓ_0 sparsity condition on a target coefficient vector, we sharpen and unify existing oracle inequalities for the Lasso and Dantzig selector. We derive oracle inequalities for target coefficient vectors with many small elements and smaller threshold levels than the universal threshold. These oracle inequalities provide sufficient conditions on the design matrix for the rate minimaxity of the Lasso and Dantzig selector for the ℓ_q risk and loss in ℓ_r balls, $0 \le r \le 1 \le q \le \infty$. By allowing $q = \infty$, our risk bounds imply the variable selection consistency of threshold Lasso and Dantzig selectors.

Keywords: variable selection, estimation, oracle inequality, minimax, linear regression, penalized least squares, linear programming

1. Introduction

As modern information technologies relentlessly generate voluminous and complex data, penalized high-dimensional regression methods have been a focus of intense research activities in machine learning and statistics in recent years. In many statistical and engineering applications, the number p of design variables (features, covariates) can be larger or even of larger order than the sample size n, but the number of important variables may still be smaller than the sample size. In such cases, one seeks a parsimonious model that fits the data well. In linear regression, a popular approach for achieving this goal is to impose a suitable penalty on the empirical loss.

This paper considers the estimation of a sparse vector of regression coefficients in a linear model. Specifically, we are interested in the rate minimaxity of the Lasso and Dantzig selector under the ℓ_q loss for the estimation of regression coefficients in ℓ_r balls. This requires lower bounds of the minimax ℓ_q risk and minimax quantiles of the ℓ_q loss over all estimators as well as matching upper bounds for the Lasso and Dantzig selector.

Let $y \in \mathbb{R}^n$ be a response vector and $X = (x_1, \ldots, x_p) \in \mathbb{R}^{n \times p}$ be a design matrix. The Lasso (Tibshirani, 1996) is the ℓ_1-penalized estimator

$$\widehat{\beta}^{(L)}(\lambda) = \arg\min_b \left\{ \|y - Xb\|^2/(2n) + \lambda \|b\|_1 \right\} \tag{1}$$

for the regression coefficients. In the signal processing literature, the Lasso is known as basis pursuit (Chen and Donoho, 1994). The Lasso has the interpretation as boosting (Freund and Schapire, 1996; Friedman, Hastie, and Tibshirani, 2000) and is computationally feasible for high-dimensional data (Osborne, Presnell, and Turlach, 2000a,b; Efron, Hastie, Johnstone, and Tibshirani, 2004). More recently, Candes and Tao (2007) proposed an ℓ_1-minimization method called the Dantzig selector,

$$\widehat{\beta}^{(D)}(\lambda) = \arg\min_b \left\{ \|b\|_1 : |x_j'(y - Xb)/n| \le \lambda, \forall j \right\}. \tag{2}$$

A focus of recent studies of high-dimensional linear regression has been on the performance of the Lasso and Dantzig selector for the estimation of the regression coefficients. Candes and Tao (2007) derived an elegant probabilistic upper bound for the ℓ_2 loss of the Dantzig selector under a condition on the number of nonzero coefficients and a uniform uncertainty principle on the Gram matrix. Efron, Hastie, and Tibshirani (2007) questioned whether a similar performance bound holds for the Lasso estimator as well. Upper bounds for the ℓ_q loss of the Lasso estimator has being studied by Bunea, Tsybakov, and Wegkamp (2007) and van de Geer (2008) for $q = 1$, Zhang and Huang (2008) for $q \in [1,2]$, Meinshausen and Yu (2009) for $q = 2$, Bickel, Ritov, and Tsybakov (2009) for $q \in [1,2]$ with a parallel analysis of the Dantzig selector, and Zhang (2009b) for $q \ge 1$. Under different sets of regularity conditions on the Gram matrix and the sparsity of regression coefficients $\beta \in \mathbb{R}^p$, these results provide error bounds of the form $\|\widehat{\beta} - \beta\|_q \le O(k^{1/q}\lambda)$, where k is the number of nonzero entries of a target vector of regression coefficients or an intrinsic dimensionality of the sparse estimation problem. For $N(0, \sigma^2)$ errors and standardized designs with $\|x_j\| = \sqrt{n}$, these studies require a universal penalty level $\lambda_{univ} = \sigma\sqrt{(2/n)\log p}$ or greater for the Dantzig selector and a penalty level λ greater by a constant factor than λ_{univ} for the Lasso. Different sets of regularity conditions lead to different forms of constant factors in the error bounds so that the existing error bounds are typically not directly comparable mathematically.

This paper contributes to high-dimensional regression in several ways. We provide lower bounds for the minimax ℓ_q risk in ℓ_r balls and the minimax quantiles of the ℓ_q loss for all designs X. We derive sufficient conditions on X for the Lasso and Dantzig selector to attain the rate of the minimax ℓ_q risk and the minimax quantiles of the ℓ_q loss. We provide oracle inequalities for the ℓ_q loss of the Lasso and Dantzig selector which sharpen, unify and extend the existing results and allow the penalty level λ to be of smaller order than the universal penalty level.

The rest of the paper is organized as follows. In Section 2, we describe lower bounds for the minimax risk and loss in ℓ_r balls. In Section 3, we provide oracle inequalities for the Lasso and Dantzig selector under the ℓ_0 sparsity of regression coefficients. We compare these oracle inequalities with existing ones and describe their implications in variable selection and rate minimaxity in ℓ_0 balls. In Section 4, we provide more general oracle inequalities to allow many small regression coefficients and penalty levels of smaller order than $\sigma\sqrt{(2/n)\log p}$. These oracle inequalities are used to establish the rate minimaxity for the ℓ_q loss in ℓ_r balls. In Section 5, we make a few remarks. An appendix contains all proofs.

We use the following notation throughout the paper. For vectors $v = (v_1, \ldots, v_p)'$, $\|v\|_0 = \#\{j : v_j \neq 0\}$ and $\|v\|_q = (\sum_j |v_j|^q)^{1/q}$ is the ℓ_q norm with the special $\|v\| = \|v\|_2$ and the usual extension to $q = \infty$. Functions are applied to vectors in individual components, $f(v) = (f(v_1), \ldots, f(v_p))'$. For matrices M and $0 \leq a, b \leq \infty$, $\|M\|_{a,b} = \max\{\|Mv\|_b : \|v\|_a = 1\}$ is the operator norm from ℓ_a to ℓ_b. For subsets A and B of $\{1, \ldots, p\}$, $X_A = (x_j, j \in A)$, $\Sigma_{A,B} = X_A' X_B / n$, $\Sigma_{A,*} = X_A' X / n$, $\Sigma_A = \Sigma_{A,A}$, and P_A is the projection from $\mathbb{R}^n$ to the linear span of $\{x_j : j \in A\}$. For real x, $x_+ = \max(x, 0)$, $1/x_+ = \infty$ for $x \leq 0$, and $\lceil x \rceil$ is the largest integer upper bound of x. For real numbers a_n and b_n, $a_n \approx b_n$ means $a_n = (1 + o(1))b_n$, $a_n \lessapprox b_n$ means $a_n \leq (1 + o(1))b_n$, and $a_n \lesssim b_n$ means $a_n = O(b_n)$. For simplicity, the dependence of estimators on the penalty level λ is suppressed unless otherwise stated.

2. Lower Bounds for the Estimation Risk and Loss

Consider the linear model

$$y = X\beta + \varepsilon = \sum_{j=1}^{p} \beta_j x_j + \varepsilon. \tag{3}$$

Throughout the sequel, $P_{\beta,X}$ is the probability measure given $\{\beta, X\}$ under which $\varepsilon \sim N(0, \sigma^2 I_n)$. For simplicity, we also assume $\|x_j\|^2 = n$ whenever $P_{\beta,X}$ is referred to. Define

$$\Theta_{r,R} = \{v \in \mathbb{R}^p : \|v\|_r \leq R\},\ r > 0, \quad \Theta_{0,k} = \{v \in \mathbb{R}^p : \|v\|_0 \leq k\}, \tag{4}$$

as the ℓ_r and ℓ_0 balls respectively. Let $\sigma_n = \sigma/\sqrt{n}$ and

$$\lambda_{univ} = \sigma_n \sqrt{2 \log p},\ \lambda_{mm} = \sigma_n \left\{ 2 \log \left(\frac{\sigma_n^r p}{R^r} \right) \right\}^{1/2},\ \lambda_{mm} = \sigma_n \sqrt{2 \log(p/k)}, \tag{5}$$

be respectively the universal (univ) penalty level (Donoho and Johnstone, 1994) and the minimax (mm) penalty levels for the ℓ_r and ℓ_0 balls. The dependence of λ_{mm} on $\{r, R\}$ or $\{0, k\}$ is suppressed since λ_{mm} is always used in association with a specific ball in (4).

Donoho and Johnstone (1994) proved that for $0 < r < q$ and based on a p-vector $\widetilde{y} \sim N(\beta, \sigma_n^2 I_p)$, the minimax ℓ_q risk in the ℓ_r ball $\Theta_{r,R}$ is approximately

$$\inf_{\delta} \sup_{\beta \in \Theta_{r,R}} E_{\beta,X} \left\| \delta(\widetilde{y}) - \beta \right\|_q^q = (1 + o(1)) R^r \lambda_{mm}^{q-r}$$

and is achieved within an infinitesimal fraction by threshold estimators at the threshold level λ_{mm}, provided that $\lambda_{mm}/\sigma_n \to \infty$ and $R^r/\lambda_{mm}^r \to \infty$. Here the infimum is taken over all Borel mappings δ of proper dimensions. The following theorem extends their result to the estimation of regression coefficients under $P_{\beta,X}$. For any class of vectors $\Theta \subset \mathbb{R}^p$, the minimax ℓ_q risk is $\mathscr{R}_q(\Theta; X) = \inf_\delta \sup_{\beta \in \Theta} E_{\beta,X} \|\delta(X, y) - \beta\|_q^q$.

Theorem 1 *Let $\Theta_{r,R}$ and $\Theta_{0,k}$ be as in (4) and λ_{mm} as in (5) with $R > 0$ and $q \geq r > 0$. Suppose $R^r/\lambda_{mm}^r \to \infty$, $k \to \infty$ and $\lambda_{mm}/\sigma_n \to \infty$. Then,*

$$\frac{\mathscr{R}_q(\Theta_{r,R}; X)}{R^r \lambda_{mm}^{q-r}} \geq (1 + o(1)), \quad \frac{\mathscr{R}_q(\Theta_{0,k}; X)}{k \lambda_{mm}^q} \geq (1 + o(1)). \tag{6}$$

Moreover, for either $\Theta = \Theta_{r,R}$ with $k = R^r/\lambda_{mm}^r$ or $\Theta = \Theta_{0,k}$,

$$\inf_X \inf_\delta \sup_{\beta \in \Theta} P_{\beta,X} \left\{ \|\delta(X, y) - \beta\|_q^q \geq (1 - \varepsilon) k \lambda_{mm}^q \right\} \geq \frac{\varepsilon + o(1)}{3^q},\ 0 < \varepsilon < 1. \tag{7}$$

Remark 2 *The value k in (7) can be viewed as an intrinsic lower bound for the number of parameters to be estimated for the minimaxity under the ℓ_q loss. By (5), $\lambda_{mm} \le \lambda_{univ}$ iff $R \ge \sigma_n$ for $r > 0$. For $\lambda_{univ} \asymp \lambda_{mm}$ and respectively for $r = 0$ and $0 < r \le 1$, Corollary 4 in Section 3.1 and Theorem 17 in Section 4.2 provide conditions on X for the Lasso and Dantzig selector to match the rate of the minimax lower bound $k\lambda_{mm}^q$ in (7). For $\lambda_{mm} \ll \lambda_{univ}$, Theorem 19 in Section 4.2 gives the rate minimaxity of the Lasso.*

During the revision of this paper, we became aware of the technical report of Raskutti, Wainwright, and Yu (2009). The lower bounds in Theorem 1 are identical for all design matrices X and thus are sharp only up to a constant factor under certain conditions on X. For example, the minimax risk in a parameter class $\Theta \ni 0$ is no smaller than the radius of the null set $\Theta \cap \{b : Xb = 0\}$. This and some other aspect of the design matrix have been used to derive sharper lower bounds in Raskutti, Wainwright, and Yu (2009). In our technical report (Ye and Zhang, 2009) and in Zhang (2009a), Theorem 1 only covers the case $r > 0$.

3. Oracle Inequalities under ℓ_0 Sparsity and Variable Selection

We discuss in three subsections oracle inequalities for the ℓ_q loss, related work, and variable selection. We focus on coefficient vectors with a relatively small number of nonzero entries here. The more complicated ℓ_r-sparse case will be considered in Section 4.

3.1 Oracle Inequalities and Rate Minimaxity under ℓ_0 Sparsity

For $\xi \ge 0$ and $J \subset \{1, \ldots, p\}$, define cone invertibility factors (CIF)

$$CIF_{q,\ell}(\xi, J) = \inf\Big\{ \frac{|J|^{1/q}\|\Sigma u\|_\infty}{\|u_A\|_q} : u \in \mathscr{C}(\xi, J), |A \setminus J| \le \ell \Big\} \tag{8}$$

with cones $\mathscr{C}(\xi, J) = \{u : \|u_{J^c}\|_1 \le \xi \|u_J\|_1 \ne 0\}$. Note that $CIF_{q,\ell}(\xi, J) = CIF_{q,p-|J|}(\xi, J)$ for $\ell \ge p - |J|$ and since the infimum is attained when u_A takes the ℓ largest elements of u outside J, $CIF_{q,\ell}(\xi, J)$ is decreasing in ℓ. Similarly, define sign-restricted cone invertibility factors (SCIF)

$$SCIF_{q,\ell}(\xi, J) = \inf\Big\{ \frac{|J|^{1/q}\|\Sigma u\|_\infty}{\|u_A\|_q} : u \in \mathscr{C}_-(\xi, J), |A \setminus J| \le \ell \Big\}. \tag{9}$$

with sign-restricted cones $\mathscr{C}_-(\xi, J) = \{u \in \mathscr{C}(\xi, J) : u_j \Sigma_{j,*} u \le 0 \ \forall j \notin J\}$. We first present oracle inequalities in terms of the (sign-restricted) CIF.

Theorem 3 *Let $\widehat{\beta}^{(D)}$ and $\widehat{\beta}^{(L)}$ be the Lasso and Dantzig selector in (1) and (2), $1 \le q \le \infty$, $\beta^* \in \mathbb{R}^p$, $J = \{j : \beta_j^* \ne 0\}$ and $z_\infty^* = \|X'(y - X\beta^*)/n\|_\infty$.*
(i) Let $CIF_{q,\ell}(1, J)$ and $SCIF_{q,\ell}(\xi, J)$ be as in (8) and (9). In the event $z_\infty^* \le \lambda$,

$$\|\widehat{\beta}^{(D)} - \beta^*\|_q \le \frac{|J|^{1/q}(\lambda + z_\infty^*)}{CIF_{q,p}(1, J)} \le \frac{2|J|^{1/q}\lambda}{CIF_{q,p}(1, J)}, \tag{10}$$

and in the event $z_\infty^* \le \lambda(\xi - 1)/(\xi + 1)$

$$\|\widehat{\beta}^{(L)} - \beta^*\|_q \le \frac{|J|^{1/q}(\lambda + z_\infty^*)}{SCIF_{q,p}(\xi, J)} \le \frac{\{2\xi/(\xi + 1)\}|J|^{1/q}\lambda}{SCIF_{q,p}(\xi, J)}. \tag{11}$$

(ii) For $\lambda = \sigma\sqrt{(2/n)\log(p/\varepsilon)}$ *and* $\lambda = \sigma\sqrt{(2/n)\log(p/\varepsilon)}(\xi+1)/(\xi-1)$ *respectively, (10) and (11) hold with* $\beta^* = \beta$ *and at least probability* $1-\varepsilon/\sqrt{\pi\log(p/\varepsilon)}$ *under* $P_{\beta,X}$.

Theorem 3 immediately provides a sufficient condition on the design X for the rate minimaxity of the Lasso and Dantzig selector in the quantiles of the ℓ_q loss in ℓ_0 balls, in the sense of (7) of Theorem 1. We state this result as the following corollary. Define

$$CIF_q^*(\xi,k) = \inf\Big\{k^{1/q}\|\Sigma u\|_\infty/\|u\|_q : 0 < \|u\|_1 \le (1+\xi)\max_{|J|=k}\|u_J\|_1\Big\}. \tag{12}$$

Corollary 4 *Suppose* $k\wedge(p/k)\to\infty$ *for certain* $(p,k) = (p_n,k_n)$.
(i) Suppose $M_{q,k}^{(D)} = 2\sqrt{\log p}/\{\sqrt{\log(p/k)}CIF_q^*(1,k)\} \le M_* < \infty$. *Then, the Dantzig selector* $\widehat\beta = \widehat\beta^{(D)}$ *with* $\lambda = \lambda_{univ}$ *in (5) is rate minimax in the sense of*

$$\sup_{\beta\in\Theta_{0,k}} P_{\beta,X}\big\{\|\widehat\beta-\beta\|_q \ge M_* k^{1/q}\lambda_{mm}\big\} \to 0. \tag{13}$$

(ii) Suppose $M_{q,k,\xi}^{(L)} = \max_{|J|\le k} 2\{\xi/(\xi-1)\}\sqrt{\log p}/\{\sqrt{\log(p/k)}SCIF_{q,p}(\xi,J)\} \le M_* < \infty$. *Then, (13) holds for the Lasso* $\widehat\beta = \widehat\beta^{(L)}$ *with* $\lambda = \lambda_{univ}(\xi+1)/(\xi-1)$.

The proof of Theorem 3 is simple since the (sign-restricted) CIF are exactly what we need. Let $h^{(D)} = \beta^{(D)}(\lambda)-\beta^*$, $h^{(L)} = \beta^{(L)}(\lambda)-\beta^*$ and $h = h^{(D)}$ or $h^{(L)}$ throughout the sequel. It follows from the feasibility of β^* for the constraint in (2) and the Karush-Kuhn-Tucker conditions for the Lasso (1) respectively that for $z_\infty^* \le \lambda$,

$$h^{(D)} \in \mathscr{C}(1,J),\ v'\Sigma h^{(L)} \le (z_\infty^*+\lambda)\|v_J\|_1 + (z_\infty^*-\lambda)\|v_{J^c}\|_1, \tag{14}$$

for all vectors v satisfying $\mathrm{sgn}(v_{J^c}) = \mathrm{sgn}(h^{(L)}_{J^c})$. With $v = h^{(L)}$ or $v_J = 0$ in (14), it follows that $h^{(L)} \in \mathscr{C}_-(\xi,J)$ for $\xi \ge (\lambda+z_\infty^*)/(\lambda-z_\infty^*)$. Theorem 3 then follows from

$$\|\Sigma h\|_\infty \le z_\infty^* + \lambda. \tag{15}$$

The lower bounds for the (sign-restricted) CIF in the following proposition facilitate more explicit results and connections to existing approaches. For given $\{J,\ell\}$ define

$$\phi_{q,\ell}(u,A,r,w,B;\xi,J) = 1 - \xi|J|^{1-1/q}(a_r/\ell)^{1-1/r}\|\Sigma_{B,A}w_A\|_{r/(r-1)} \tag{16}$$

in the following domain: $u\in\mathscr{C}(\xi,J)$, $A = \arg\max_{|B\setminus J|\le\ell}\|u_B\|_1$ (determined by u up to ties), $r\ge 1$, $w_A'\Sigma_A u_A = \|u_A\|_q$, $B\cap A = \emptyset$, $|B| = \lceil \ell/a_r\rceil$ and $a_r = (1-1/r)/r^{1/(r-1)}$.

Proposition 5 *Let* $\{\xi,q,r\}\subset[1,\infty]$, $1\le s\le q$ *and* $0<\ell\le p-|J|$. *Then,*

$$\max\Big\{\frac{CIF_{q,\ell}(\xi,J)}{CIF_{s,p}(\xi,J)}, \frac{SCIF_{q,\ell}(\xi,J)}{SCIF_{s,p}(\xi,J)}\Big\} \le C_{s,q}(\xi,|J|/\ell), \tag{17}$$

where $C_{s,q}(\xi,t) = (1+\xi)^{(1/s-1/q)/(1-1/q)}\{1+\xi^q(a_q t)^{q-1}\}^{(1-1/s)/(q-1)}$;

$$CIF_{q,\ell}(\xi,J) = \inf_{u\in\mathscr{C}(\xi,J)}\inf_{|A\setminus J|=\ell}\sup_{v\ne0}\frac{|J|^{1/q}v'\Sigma u}{\|v\|_1\|u_A\|_q} \ge \phi_{q,\ell}^*(\xi,J), \tag{18}$$

where $\phi^*_{q,\ell}(\xi,J) = \min_{u,A}\max_{r,w}\min_B \phi_{q,\ell}(u,A,r,w,B;\xi,J)|J|^{1/q}/\|w_A\|_1$*; and*

$$SCIF_{q,\ell}(\xi,J) \geq \inf_{u\in\mathscr{C}_-(\xi,J)}\inf_{|A\setminus J|=\ell}\sup_{v\in\mathscr{Q}(u,J)}\frac{|J|^{1/q}v'\Sigma u}{\|v_J\|_1\|u_A\|_q} \geq \phi^{**}_{q,\ell}(\xi,J), \tag{19}$$

where $\phi^{**}_{q,\ell}(\xi,J) = \min_{u,A}\max_{r,w}\min_B \phi_{q,\ell}(u,A,r,w,B;\xi,J)|J|^{1/q}/\|w_J\|_1$ *with* $\mathscr{Q}(u,J) = \{v : sgn(v_{J^c}) = sgn(u_{J^c})\}$ *and vectors* $u \in \mathscr{C}_-(\xi,J)$ *and* $w \in \mathscr{Q}(u,J)$.

Remark 6 *Taking* $w = u\|u_A\|/(u_A'\Sigma_A u_A)$ *in (16) for the* ℓ_2 *loss, we find*

$$\begin{aligned}\widetilde{\phi}^*_{2,\ell}(\xi,J) &= \min_{u,A}\max_r\min_B\big\{u_A'\Sigma_A u_A - \xi|J|^{1/2}(a_r/\ell)^{1-1/r}\|\Sigma_{B,A}u_A\|_{r/(r-1)}\big\},\\ &\leq \min\big\{\phi^{**}_{2,\ell}(\xi,J), \{(1+\ell/|J|)^{1/2}\wedge(1+\xi)\}\phi^*_{2,\ell}(\xi,J)\big\}\end{aligned} \tag{20}$$

with $\|u_A\| = 1$ *and* $\{A,r,B\}$ *in (16), due to* $\|u_A\|_1 \leq \{(1+\ell/|J|)^{1/2}\wedge(1+\xi)\}|J|^{1/2}$ *and* $\|w_J\|_1 \leq |J|^{1/2}/(u_A'\Sigma_A u_A)$*. However, for* $q > 2$*,* $w \propto u$ *in (16) does not lead to a right normalization since* $\|u_A\|/|A|^{1/2}$ *does not control* $\|u_A\|_q/|A|^{1/q}$*. For general* $q \in [1,\infty]$*,* $\|\Sigma_A w_A\|_{q/(q-1)} = w_A'\Sigma_A u_A/\|u_A\|_q = 1$ *gives a properly length normalized lower bound:*

$$\phi^*_{q,\ell}(\xi,J) \geq \min_{A,B}\big\{1-\xi|J|^{1-1/q}(a_r/\ell)^{1-1/r}\|\Sigma_A^{-1}\Sigma_{A,B}\|_{r,q}\big\}|J|^{1/q}/\|\Sigma_A^{-1}\|_{\infty,q}. \tag{21}$$

For $(|A|,|B|,\|u\|,\|v\|_r) = (\lceil a\rceil,\lceil b\rceil,1,1)$ with $A\cap B = \emptyset$, define

$$\delta_a^{\pm} = \max_{A,u}\Big\{\pm\Big(\|\Sigma_A u\| - 1\Big)\Big\},\ \delta_a = \delta_a^+\vee\delta_a^-,\ \theta^{(r)}_{a,b} = \max_{A,B,u,v} v'\Sigma_{A,B}u. \tag{22}$$

Oracle inequalities for the ℓ_q loss, $1\leq q\leq 2$, have been given in terms of the quantities in (22). The quantities $1\pm\delta_a^{\pm}$ give the maximum and minimum sparse eigenvalues of the Gram matrix Σ up to dimension $\lceil a\rceil$. For $\|u_A\| = 1$, we have $u_A'\Sigma_A u_A \geq 1-\delta^-_{|A|}$, $\|\Sigma_{B,A}u_A\|_{r/(r-1)} \leq \theta^{(r)}_{|B|,|A|}$ and $\|\Sigma_{B,A}u_A\|^2 \leq (1+\delta^+_{|B|})u_A'\Sigma_A u_A$. Thus, Theorem 3, (17) with $C_{q,2}(\xi,k/\ell) = (1+\xi)^{2/q-1}\{1+\xi^2 k/(4\ell)\}^{1-1/q}$, and (20) with $r\in\{2,\infty\}$ yield the following corollary.

Corollary 7 *Let* $\|\beta^*\|_0 = |J| = k$ *and* $1\leq q\leq 2$*. Then, for* $z^*_\infty = \|X'(y-X\beta^*)/n\|_\infty \leq \lambda$*,*

$$\|\widehat{\beta}^{(D)}(\lambda)-\beta^*\|_q \leq \frac{2|J|^{1/q}\lambda}{CIF_{q,p}(1,J)} \leq \frac{2^{2/q}(1+k/(4\ell))^{1-1/q}(2\wedge\sqrt{1+\ell/k})k^{1/q}\lambda}{(1-\delta^-_{k+\ell})_+\{1-\min(\widetilde{F}_1,\widetilde{F}_2,\widetilde{F}_3)\}_+}, \tag{23}$$

where $1\leq\ell\leq p-k$*,* $\widetilde{F}_1 = (k^{1/2}/\ell)\theta^{(\infty)}_{\ell,k+\ell}/(1-\delta^-_{k+\ell})_+$*,* $\widetilde{F}_2 = \sqrt{k/(4\ell)}\,\theta^{(2)}_{4\ell,k+\ell}/(1-\delta^-_{k+\ell})_+$ *and* $\widetilde{F}_3 = \{(k/(4\ell))(1+\delta^+_{4\ell})/(1-\delta^-_{k+\ell})_+\}^{1/2}$*. Moreover, for* $z^*_\infty \leq \lambda(\xi-1)/(\xi+1)$*,*

$$\|\widehat{\beta}^{(L)}(\lambda)-\beta^*\|_q \leq \frac{\{2\xi/(\xi+1)\}|J|^{1/q}\lambda}{SCIF_{q,p}(\xi,J)} \leq \frac{(1+\xi)^{2/q-2}(1+\xi^2 k/(4\ell))^{1-1/q}2\xi k^{1/q}\lambda}{(1-\delta^-_{k+\ell})_+\{1-\xi\min(\widetilde{F}_1,\widetilde{F}_2,\widetilde{F}_3)\}_+}. \tag{24}$$

3.2 Related Work

Here we discuss the connections between the results in Section 3.1 and related work. We show that the (sign-restricted) CIF-based oracle inequalities in Theorem 3 and Corollary 7 sharpen, unify and extend existing performance bounds and offer a clear explanation of the differences among existing analyses.

The CIF and SCIF in (8) and (9) are related to the restricted eigenvalues (RE)

$$RE_{q,\ell}(\xi,J) = \inf\Big\{ \frac{|J|^{1/q}\|Xu\|}{|nJ|^{1/2}\|u_A\|_q} : u \in \mathscr{C}(\xi,J), |A \setminus J| \le \ell \Big\}, \tag{25}$$

since they all conveniently provide factors required in proofs of the same type of oracle inequalities. The quantity (25) includes as special cases the ℓ_2 version $RE_{2,p}(\xi,J)$ of Bickel, Ritov, and Tsybakov (2009) and Koltchinskii (2009) and the constant factor $RE_{1,0}(3,J)$ for the compatibility condition of van de Geer (2007). van de Geer and Bühlmann (2009) called $RE_{1,0}(\xi,J)$ the restricted ℓ_1-eigenvalue.

Both types of quantities in (8), (9) and (25) involve minimum ratios of seminorms over cones. The main differences between them are the quantities used to bound the estimation error and the sign-restriction for $SCIF_{q,\ell}(\xi,J)$. Let $\{h^{(L)}, h^{(D)}, h\}$ be as in (14) and (15). While the RE in (25) uses the prediction regret $h'\Sigma h$ to bound the estimation error h, the (sign-restricted) CIF in (8) and (9) uses the ℓ_∞ norm of the gradient to bound the estimation error via (15). The use of the gradient bound in (8) and (9) provides sharper oracle inequalities by allowing the flexibility with the choice of v in (18) and (19).

For the ℓ_2 loss of the Dantzig selector, the oracle inequality of Bickel, Ritov, and Tsybakov (2009) and Theorem 3 can be compared as follows: for $z^* \le \lambda$

$$\|h^{(D)}\| \le \frac{2|J|^{1/2}\lambda}{CIF_{2,p}(1,J)} \le \frac{4|J|^{1/2}\lambda}{RE_{1,0}(1,J)RE_{2,p}(1,J)} \le \frac{4(1+\sqrt{|J|/\ell})|J|^{1/2}\lambda}{RE^2_{2,\ell}(1,J)}. \tag{26}$$

The right-hand side of (26) is the upper bound in (7.6) of Theorem 7.1 of Bickel, Ritov, and Tsybakov (2009). An application of the upper bound of van de Geer and Bühlmann (2009) on $\|h^{(D)}_J\|_1$ in a modification of the proof of Bickel, Ritov, and Tsybakov (2009) yields the intermediate upper bound. Theorem 3 provides the sharpest upper bound in (26). The second inequality in (26) follows from (18) with $\{\ell, v\} = \{p-|J|, u\}$. The third inequality in (26) follows from $RE_{1,0}(\xi,J) \ge RE_{2,\ell}(\xi,J)$ as in van de Geer and Bühlmann (2009) and $(1+\xi|J|/\ell)^{1/2}RE_{2,p}(\xi,J) \ge RE_{2,\ell}(\xi,J)$ by the shifting inequality in Candes and Tao (2007). The shifting inequality of Cai, Wang, and Xu (2010) can be used to reduce the factor $(1+\sqrt{|J|/\ell})$ to $\sqrt{1+|J|/(4\ell)}$ in (26), but (26) still holds. Similarly, for $z^* \le \lambda/2$,

$$\|h^{(L)}\| \le \frac{(3/2)|J|^{1/2}\lambda}{SCIF_{2,p}(3,J)} \le \frac{(3/2)|J|^{1/2}\lambda}{RE_{1,0}(3,J)RE_{2,p}(3,J)} \le \frac{4(1+3\sqrt{|J|/\ell})|J|^{1/2}\lambda}{RE^2_{2,\ell}(3,J)}, \tag{27}$$

with (7.10) of Theorem 7.2 of Bickel, Ritov, and Tsybakov (2009) on the right-hand side. The differences of the upper bounds in (26) and (27) could be nontrivial since van de Geer and Bühlmann (2009) showed by example the possibility of $RE_{2,0}(\xi,J)/RE_{1,0}(\xi,J) \to 0$.

A significant difference between the (sign-restricted) CIF- and RE- based oracle inequalities is their relationships to the oracle inequalities of Candes and Tao (2007) and Zhang (2009b). While

Theorem 3 is sharper than these inequalities, the RE-based oracle inequalities seem not to have this property.

The flexibility with the choice of v in (18) and (19) is a significant advantage for (8) and (9). The square of the potentially small $(\|\Sigma_A^{1/2}u_A\| - \|\Sigma_{A^c}^{1/2}u_{A^c}\|)_+$ has been used to bound $u'\Sigma u = \|Xu\|^2/n$ from below in the proofs of RE-based oracle inequalities. This is not needed in Candes and Tao (2007) and Zhang (2009b) since their arguments correspond to using vectors v with $v_{A^c} = 0$ in (18) and (19). For example, (23) with $\widetilde{F}_3$ is at least by a factor $\{1 - \min(\widetilde{F}_1, \widetilde{F}_2, \widetilde{F}_3)\}_+$ sharper than inserting Lemma 4.1 (ii) into (7.6) in Bickel, Ritov, and Tsybakov (2009), even after an application of sharper shifting inequalities to their Lemma 4.1 (ii). For $q > 2$, the ratio in (18) is not properly length normalized with $v = u$, as discussed in Remark 6. Thus, direct extensions of (25) with $u'\Sigma u$ in the numerator may not yield performance bounds of the right order.

Corollary 7 sharpens the oracle inequality of Candes and Tao (2007). The upper bound (23) with $\widetilde{F}_1$ is of the sharpest form among the three for large ℓ due to the factor $1/\ell$, compared with $\sqrt{1/(4\ell)}$ for $\widetilde{F}_2$ and $\widetilde{F}_3$. For $\ell = k/4$ and $q = 2$, (23) with $\widetilde{F}_2$ yields

$$z^* \le \lambda \;\Rightarrow\; \|\widehat{\beta}^{(D)}(\lambda) - \beta^*\| \le \frac{2\sqrt{k}\lambda}{CIF_{2,p}(1,J)} \le \frac{\sqrt{10k}\lambda}{(1 - \delta^-_{1.25k} - \theta^{(2)}_{k,1.25k})_+}, \tag{28}$$

a slightly sharper version of Cai, Wang, and Xu (2010) improvement of Candes and Tao (2007) result due to $\delta^-_{1.25k} \le \delta_{1.25k}$. Similarly, (24) with $\xi = \sqrt{2}$ and $\ell = k/2$ yields

$$z^* \le \lambda \;\Rightarrow\; \|\widehat{\beta}^{(L)}\big((1+\sqrt{2})^2\lambda\big) - \beta^*\| \le \frac{(1+\sqrt{2})\sqrt{8k}\lambda}{SCIF_{2,p}(\sqrt{2},J)} \le \frac{(1+\sqrt{2})4\sqrt{k}\lambda}{(1 - \delta^-_{1.5k} - \theta^{(2)}_{2k,1.5k})_+}. \tag{29}$$

The right-hand sides of (28) and (29) are directly comparable with the Candes and Tao (2007) inequality $\|\widehat{\beta}^{(D)}(\lambda) - \beta^*\| \le 4\sqrt{k}\lambda/(1 - \delta_{2k} - \theta^{(2)}_{k,2k})_+$ in the same event.

Another option is to apply (17), (18) and (21) with $(r,s) = (\infty, q)$ to (10), resulting in

$$\|\widehat{\beta}^{(D)}(\lambda) - \beta^*\|_q \le \frac{2|J|^{1/q}\lambda}{CIF_{q,p}(1,J)} \le \max_{A,B} \frac{2\lambda\|\Sigma_A^{-1}\|_{\infty,q}(1 + (a_q k/\ell)^{q-1})^{1/q}}{\{1 - (k^{1-1/q}/\ell)\|\Sigma_A^{-1}\Sigma_{A,B}\|_{\infty,q}\}_+} \tag{30}$$

for all $1 \le q \le \infty$ in the event $z^*_\infty \le \lambda$, where $k = |J|$ and A and B are as in (16). Inequality (30) and the combination of (11) and (19) with $w_A \propto \mathrm{sgn}(u_A)|u_A|^{q-1}$ are related to the results of Zhang (2009b). For $\|\beta^*\|_0 = k$, his oracle inequality for the Lasso can be written as

$$\big\|\widehat{\beta}^{(L)}(\lambda/t^*) - \beta^*\big\|_q \le \frac{32(1+F^*)G^*}{\widetilde{C}(1-F^*)_+^2} \quad \text{for } z^*_\infty \le \lambda,\ 1 \le q \le \infty,\ \ell \ge k, \tag{31}$$

with $F^* = \max_{A,B,w}(k^{1-1/q}\|\Sigma_{B,A}w_A\|_1/\ell)$, $G^* = \widetilde{C}\lambda(k+\ell)^{1/q}\max_{A,w}\|w_A\|_{q/(q-1)}$ for $w \propto \Sigma_A^{-1}\mathrm{sgn}(u_A)|u_A|^{q-1}$, $G^* = \widetilde{C}\lambda k^{1/q}\max_{A,w}\|w_A\|_{q/(q-1)}$ for $w_A \propto \mathrm{sgn}(u_A)|u_A|^{q-1}$, $\widetilde{C} = (1 + 1/t^*)$ and $t^* = (1-F^*)/\{4(1+F^*)\}$, where $w_A'\Sigma_A u_A = \|u_A\|_q = 1$ and $\{A, B, u_A\}$ are as in (16). It turns out that the combination of (11) and (19) with the same choice of w and $(r, a_r) = (\infty, 1)$ is at least by a factor of $5/12$ sharper than (31), even with the suboptimal $a_q = 1$. For small k/ℓ, Zhang (2009b) pointed out the smaller order $k^{1/q}\lambda$ of G^* for $w_A \propto \mathrm{sgn}(u_A)|u_A|^{q-1}$ as an advantage of the Lasso, compared with the order $(k+\ell)^{1/q}\lambda$. The cost of this advantage is the square of $(1-F^*)_+$ in the

denominator of (31), compared with (23) and (30) for the Dantzig selector. Moreover, the error bound in (23) for the Dantzig selector is also of the order $k^{1/q}\lambda$ for $q \le 2$ with much smaller constant factors, and the difference between $k^{1/q}$ and $(k+\ell)^{1/q}$ diminishes for large q as in Theorem 8. Thus, the advantage of the Lasso in this aspect has some limitations.

We have proved that Theorem 3 sharpens and unifies a number of existing oracle inequalities for the Lasso and Dantzig estimators, so that they can all be viewed as (possibly more explicit) upper bounds for the right-hand sides of (10) and (11). The choice $r = \infty$ in (16), and consequently in (18), (19), (20) and (21), typically gives oracle inequalities of the sharpest form involving the dimension-normalized $\|\cdot\|_{\infty,q}$ norm as in (30), compared with the typical $\|\cdot\|_{q,q}$ norm in the literature. Oracle inequalities for $\beta^*_{J^c} \ne 0$ are given in Section 4. Although (10) and (11) are of the same format, the Dantzig selector requires smaller λ and smaller $\xi = 1$. This theoretical advantage of the Dantzig selector for $z^*_\infty \le \lambda$ reverses if $\|\widehat{\beta}^{(D)}(\lambda)\|_1 \le \|\beta^*\|_1$ is replaced by $\|\widehat{\beta}^{(D)}(\lambda)\|_1 \le \|\widehat{\beta}^{(L)}(\lambda)\|_1$ for $z^*_\infty > \lambda$.

3.3 Vaiable Selection

Variable selection is fundamental for the interpretation of models with hign-dimensional data. Meinshausen and Bühlmann (2006), Tropp (2006), Zhao and Yu (2006), and Wainwright (2009) proved that the Lasso is variable selection consistent under a strong irrepresentable condition and some other regularity conditions on X and β. Candes and Plan (2009) proved the selection consistency of the Lasso under random permutation and sign-change of regression coefficients and a mild condition on the maximum absolute correlation among design vectors. Consistent variable selection in linear regression can be achieved with a concave penalty (Fan and Li, 2001; Zhang, 2010) or adaptive Lasso (Zou, 2006; Huang, Ma, and Zhang, 2008), without requiring the strong irrepresentable condition.

The ℓ_∞ error bounds in Theorem 3 (i) and their more explicit versions, for example, (30) with $q = \infty$, naturally lead to variable selection by thresholding the Lasso or Dantzig selector. We focus on the Dantzig selector here although parallel results can be obtained for the Lasso in the same way. Zhang (2009b) studied the selection consistency of thresholding the Lasso through his upper bounds for $\|\widehat{\beta}^{(L)} - \beta^*\|_\infty$. Lounici (2008) considered thresholding either the Lasso or the Dantzig selector under the stronger condition $\max_{j\ne k}|(\Sigma)_{jk}| \le 1/\{\alpha(1+2\xi)|J|\}$ with $\alpha > 1$, $(\Sigma)_{jj} = 1$, $\xi = 3$ for the Lasso, and $\xi = 1$ for the Dantzig selector.

Candes and Tao (2007) considered the Gauss-Dantzig selector

$$\widehat{\beta}^{(GD)} = \arg\min_b \Big\{ \|y - Xb\| : |\widehat{\beta}_j| \le \lambda' \Rightarrow b_j = 0, \forall j, \widehat{\beta} = \widehat{\beta}^{(D)}(\lambda) \Big\}. \tag{32}$$

For threshold functions $t(x;\lambda)$ satisfying $\{x : t(x;\lambda) = 0\} = \{x : |x| \le \lambda\}$ and $xt(x;\lambda) \ge 0$, define the threshold Dantzig selector as

$$\widehat{\beta}^{(TD)} = t\big(\widehat{\beta}^{(D)}(\lambda);\lambda'\big). \tag{33}$$

This includes the hard $t(x;\lambda) = xI\{|x| > \lambda\}$ and the soft $t(x;\lambda) = \mathrm{sgn}(x)(|x| - \lambda)_+$. Define

$$\widehat{\beta}^{(oracle)} = \arg\min_b \Big\{ \|y - Xb\| : \beta_j = 0 \Rightarrow b_j = 0, \forall j \Big\}.$$

Theorem 8 *Suppose (3) holds with $J = \{j : \beta_j \neq 0\}$. Let $\widehat{\beta}^{(GD)}$ and $\widehat{\beta}^{(TD)}$ be as in (32) and (33) respectively with the universal penalty level $\lambda = \lambda_{univ} = \sigma\sqrt{(2/n)\log p}$ and a threshold level λ' satisfying $2\lambda_{univ}/CIF_{\infty,p}(1,J) \leq \lambda' < \min_{\beta_j \neq 0} |\beta_j|/2$. Then,*

$$P_{\beta,X}\Big\{ sgn(\widehat{\beta}^{(TD)}) \neq sgn(\beta) \text{ or } \widehat{\beta}^{(GD)} \neq \widehat{\beta}^{(oracle)} \Big\} \leq 1/\sqrt{\pi \log p} \to 0. \tag{34}$$

Remark 9 *If we use (21) in Theorem 8, $(|J|/\ell)\max_{A,B} \|\Sigma_A^{-1}\Sigma_{A,B}\|_{\infty,\infty} < 1$ becomes a basic condition for (34). Meanwhile, the strong irrepresentable condition for the selection consistency of the Lasso without post-thresholding is $\|\Sigma_{J^c,J}\Sigma_J^{-1}\|_{\infty,\infty} < 1$. Compared with Lounici (2008), we improve the factor $1+2\xi$ to $1+\xi$ via*

$$CIF_{\infty,p}(\xi,J) \geq \min_{j\in J}(\Sigma)_{jj} - (|J|-1+\xi|J|)\max_{j\neq k}|(\Sigma)_{jk}|.$$

4. Upper Bounds for the ℓ_q Loss in ℓ_r Balls

We divide this section into two subsections. The first subsection provides non-probabilistic oracle inequalities: conditions on the data (X,y) and a target coefficient vector β^* for upper bounds of $\|\widehat{\beta} - \beta^*\|_q$ for the Lasso and Dantzig selector. The second subsection provides sufficient condition on the design X for the rate minimaxity for the ℓ_q loss in ℓ_r balls under $P_{\beta,X}$.

4.1 Oracle Inequalities

The oracle inequalities here differ from Theorem 3 (i) by allowing target vectors with many small entries and smaller penalty levels.

Our first theorem deals with the usual $\lambda \geq z_\infty^*$ and allows targets β^* with small $\|\beta^*_{J^c}\|_1$ for a certain set $J \subset \{1,\ldots,p\}$. The effect of the elements of β^* in J^c is controlled by

$$M_q(\lambda,\rho) = \sup\Big\{ \|u\|_q : \|\Sigma u\|_\infty \leq \lambda, \|u\|_1 \leq \rho \Big\}. \tag{35}$$

Theorem 10 *Let β^* be a target vector, $q \in [1,\infty]$, $J \subset \{1,\ldots,p\}$ and $\rho_J = \|\beta^*_{J^c}\|_1$. Then,*

$$\|\widehat{\beta}^{(D)} - \beta^*\|_q \leq \max\Big\{ \frac{2|J|^{1/q}\lambda}{CIF_{q,p}(\xi,J)}, 2M_q\Big(\lambda, \frac{\xi+1}{\xi-1}\rho_J\Big)\Big\}, \ \forall \xi > 1, \tag{36}$$

in the event $z_\infty^* = \|X'(y - X\beta^*)/n\|_\infty \leq \lambda$. Moreover, for $z_\infty^* \leq \lambda(\xi_0-1)/(\xi_0+1)$,

$$\|\widehat{\beta}^{(L)} - \beta^*\|_q \leq \max\Big\{ \frac{\xi_1|J|^{1/q}\lambda}{CIF_{q,p}(\xi,J)}, M_q\big(\xi_1\lambda, \xi_2\rho_J\big)\Big\}, \ \forall \xi > \xi_0, \tag{37}$$

where $CIF_{q,\ell}(\xi,J)$ is as in (8), $\xi_1 = 2\xi_0/(\xi_0+1)$ and $\xi_2 = (1+\xi_0)(1+\xi)/(\xi-\xi_0)$.

Remark 11 *The first component of (36) and (37) can be viewed as the cost of estimating the large components β^* in J without knowing J, and the second component the cost of having potentially many small elements of β^* in J^c. Since $M_q(\lambda,\rho) \leq \rho$ for $q \geq 1$, $\beta^*_{J^c}$ does not contribute to the order of the error bounds in (36) and (37) when $\rho_J \lesssim |J|^{1/q}\lambda/CIF_{q,p}(\xi,J)$. In Proposition 15 below, we provide conditions for $M_q(\lambda,\rho) \lesssim \lambda(\rho/\lambda)^{1/q}$, as if $\Sigma = I$.*

Our next theorem deals with smaller penalty levels satisfying $z^*_{2,d} \le \lambda < z^*_\infty$, where

$$z^*_{q,d} = \max_{|A|=d} z^*_{q,A},\ z^*_{q,A} = \|X'_A(y - X\beta^*)/n\|_q/|A|^{1/q}, \quad z^*_\infty = z^*_{\infty,1}. \tag{38}$$

Since $z^*_{q,d}$ is the length normalized ℓ_q norm of the d largest absolute values of the elements of $z = X'(y - X\beta^*)/n$, $z^*_{q,d}$ is increasing in q and decreasing in d, and $z^*_{q,d} \le z^*_\infty$. Let

$$\mathscr{C}_{relax}(\xi, J, d) = \Big\{ u : \|u_{J^c}\|_1 \le \xi d^{1/2} \max_{|A|=d, A\supseteq J} \|u_A\| \Big\} \tag{39}$$

as a relaxed cone, and define the corresponding relaxed CIF as

$$CIF_{q,relax}(\xi, J, d) = \inf_{u\in\mathscr{C}_{relax}(\xi,J,d)} \Big\{ \frac{d^{1/q}\|\Sigma_{A,*}u\|}{d^{1/2}\|u\|_q} : A = \operatorname*{arg\,max}_{|B|=d, B\supseteq J} \|u_B\| \Big\}. \tag{40}$$

The following quantity plays the role of (35) for relaxed cones:

$$M_{q,relax}(\lambda, \rho, J, d) = \sup_{\|u\|_1\le\rho} \Big\{ \|u\|_q : \|\Sigma_{A,*}u\| \le d^{1/2}\lambda, A = \operatorname*{arg\,max}_{|B|=d, B\supseteq J} \|u_B\| \Big\}. \tag{41}$$

Since $\|u_J\|_1 \le |A|^{1/2}\|u_A\|$ for $A \supseteq J$, the relaxed cone (39) is larger than the cone in (8). Moreover, since $\|\Sigma_{A,*}u\|/|A|^{1/2} \le \|\Sigma u\|_\infty$,

$$CIF_{q,relax}(\xi, J, d) \le (d/|J|)^{1/q} CIF_{q,p}(\xi, J),\ M_{q,relax}(\lambda, \rho, J, d) \ge M_q(\lambda, \rho).$$

Theorem 12 *Let β^* be a target vector, $q \in [1,\infty]$, $J \subset \{1,\dots,p\}$ with $(4|J|/3) \vee 1 \le d \le p$, $\rho_J = \|\beta^*_{J^c}\|_1$ and $z^*_{2,d}$ be as in (38). Then, for $z^*_{2,d} \le \lambda(\xi_0 - 1)/(\xi_0 + 1)$,*

$$\|\widehat{\beta}^{(L)} - \beta^*\|_q \le \max\Big\{ \frac{\xi_1 d^{1/q}\lambda}{CIF_{q,relax}(\xi, J, d)}, M_{q,relax}\Big(\xi_1\lambda, \xi_2\rho_J, J, d\Big) \Big\}, \tag{42}$$

where $\xi_1 = 2\xi_0/(\xi_0+1)$ and $\xi_2 = (\xi_0+1)(\xi+1)/(\xi-\xi_0)$.

Remark 13 *By (22) and (38), $(z^*_{2,d})^2/\{\sigma^2(1+\delta^+_d)\}$ is no greater than the maximum of $\binom{p}{d}$ χ^2_d variables under $P_{\beta^*,X}$ in (3), so that for certain $\lambda \asymp \sigma\{(1+\delta^+_d)(2/n)\log(p/d)\}^{1/2}$, $z^*_{2,d} \le \lambda(\xi_0 - 1)/(\xi_0+1)$ with large probability. Thus, for $|J| = k \asymp d$ and $\log(p/d) \ll \log p$, Theorem 12 allows $\lambda \ll \lambda_{univ} = \sigma\{(2/n)\log p\}^{1/2}$. Zhang (2010) derived similar oracle inequalities for the Lasso, MC+, and other concave penalized least squares estimators at the same λ under the sparse Riesz condition $|J| \le d/\{(1+\delta^+_d)/(1-\delta^-_d) + 1/2\}$.*

Remark 14 *Since $\|u_A\|/d^{1/2}$ does not control $\|u_A\|_q/d^{1/q}$ for $q > 2$ and large $|A| = d$, the relaxed constant factors in (40) and (41) are not properly normalized for $q > 2$ and $\Sigma = I$. Thus, Theorem 12 is most useful when $1 \le q \le 2$, although it is valid for all $1 \le q \le \infty$.*

We use the following quantities to bound the constant factors in Theorems 10 and 12:

$$\begin{aligned} \eta_{q,d} &= \max_{|A|=d} \|\Sigma_A^{-1}\|_{\infty,q}/d^{1/q},\ \eta^*_{q,d} = \max_A \|\Sigma_A^{-1}\|_{q,q}, \\ \kappa_{q,d,\ell} &= \max_{|A|=d} \min_{r\ge1} \max_{|B|=\ell} \ell^{1-1/q}(a_r/\ell)^{1-1/r}\|\Sigma_A^{-1}\Sigma_{A,B}\|_{r,q}. \end{aligned} \tag{43}$$

Proposition 15 *Let $CIF_{q,relax}(\xi,J,\ell)$, $M_q(\lambda,\rho)$ and $M_{q,relax}(\lambda,\rho,J,\ell)$ be as in (40), (35) and (41) respectively. Let $\{\eta_{q,d},\eta^*_{q,d},\kappa_{q,d,\ell}\}$ be as in (43) and $a_q=(1-1/q)/q^{1/(q-1)}$. Then,*

$$M_q(\lambda,\rho)\le \eta_{q,k}k^{1/q}\lambda+(\kappa^q_{q,k,k}+a_q^{q-1})^{1/q}k^{1/q-1}\rho,\ \forall\, k\ge 1, 1\le q\le\infty, \tag{44}$$

$$M_{q,relax}(\lambda,\rho,J,d)\le \eta^*_{2,d}d^{1/q}\lambda+\{(d/\ell)^{1-q/2}\kappa^q_{2,d,\ell}+a_q^{q-1}\}^{1/q}\ell^{1/q-1}\rho,\ 1\le q\le 2, \tag{45}$$

with $\ell=d-|J|$, and with $C_{q,2}(\xi,t)$ and $\widetilde{\phi}^*_{2,\ell}(\xi,J)$ as in (17) and (20),

$$CIF_{q,p}(\xi,J)\ge CIF_{q,relax}(\xi,J,\ell)\ge \frac{(d/|J|)^{1/q-1/2}\widetilde{\phi}^*_{2,\ell}(\xi\sqrt{d/|J|},J)}{C_{q,2}\big(\xi(d/|J|)^{1/2},|J|^2/(d\ell)\big)}. \tag{46}$$

Remark 16 *Suppose that the quantities in (43) are bounded whenever invoked. For $\rho/\lambda\asymp k\asymp\ell$, (44) and (45) give the rate $M_q(\lambda,\rho)\lesssim\rho(\lambda/\rho)^{1-1/q}$ and $M_{q,relax}(\lambda,\rho,J,d)\lesssim\rho(\lambda/\rho)^{1-1/q}$, the same as the simplest case $\Sigma=I$. Since (46) is of the form (20), Corollary 7 can be automatically extended under the setting of Theorem 12.*

4.2 Rate ℓ_q Minimax Estimation in ℓ_r Balls

We present sufficient conditions for the rate minimaxity of the Lasso and Dantzig selector in ℓ_r balls in (4) in the sense of (6) and (7). Let λ_{univ} and λ_{mm} be as in (5). We first consider the ℓ_q risk.

Theorem 17 *Let $q\ge 1\ge r>0$. Suppose $(\log p)/n=O(1)$ and $R^r/\lambda^r_{mm}\asymp d\le n\wedge p$ for some integer $d\to\infty$ satisfying $(\log d)/\log p\le c_0<1$. Let $0<\alpha_0<1$ and $\widehat{\beta}$ be either the Lasso or Dantzig selector with $\lambda=\lambda_{univ}/\alpha_0$. Suppose $p^{1-(\alpha_1/\alpha_0)^2}(n^q/d+d^{q/r-1})\le 1$ and $p^{1-(\alpha/\alpha_0)^2}d^{q/r-q}\le 1$ for certain $\{\alpha,\alpha_1\}\subset(\alpha_0,1)$. For the Dantzig selector, let $\xi>1$, $\xi^*=1$ for $r=1$ and $\xi^*=(\xi+1)/(\xi-1)$ for $r<1$. For the Lasso, let $\xi>(1+\alpha)/(1-\alpha)$ $\xi^*=1/(1-\alpha_1)$ for $r=1$ and $\xi^*=(\xi+1)/\{\xi-1-\alpha(\xi+1)\}$ for $r<1$. Then,*

$$\frac{\sup_{\beta\in\Theta_{r,R}}E_{\beta,X}\|\widehat{\beta}-\beta\|_q^q}{R^r\lambda_{mm}^{q-r}}\lessapprox\Big[\max\Big\{\frac{C_1I\{r<1\}}{CIF_q^*(\xi,d)},C_1M_q\Big(\frac{1}{d^{1/q}},\frac{\xi^*C_2}{d^{1/q-1}}\Big)\Big\}\Big]^q, \tag{47}$$

where $C_1=2(d\lambda^r_{mm}/R^r)^{1/q}/(\alpha_0\sqrt{1-c_0})$, $C_2=\alpha_0\sqrt{1-c_0}R/(d^{1/r}\lambda_{mm})$, $CIF_q^*(\xi,d)$ is as in (12) and $M_q(\lambda,\rho)$ is as in (35). Consequently, if either $\eta_{q,d}+\kappa_{q,d,d}=O(1)$ with the $\{\eta_{q,d},\kappa_{q,d,d}\}$ in (43) or $M_q(d^{-1/q},d^{1-1/q})+I\{r<1\}/CIF_q^*(\xi,d)=O(1)$, then

$$\sup_{\|\beta\|_r\le R}E_{\beta,X}\|\widehat{\beta}-\beta\|_q^q\lesssim\inf_{\delta}\sup_{\|\beta\|_r\le R}E_{\beta,X}\|\delta(X,y)-\beta\|_q^q. \tag{48}$$

Remark 18 *For $q=2$, $\eta_{q,k}+\kappa_{q,k,k}=O(1)$ for some $k\asymp d$ if the sparse Riesz condition holds Zhang and Huang (2008), that is, $1/(1-\delta_d^-)+\delta_d^+=O(1)$ for the $\delta_d^\pm$ in (22). For $p\gg n$, random matrix theory can be applied to validate such conditions up to $d\asymp n/\log(p/n)$.*

Theorem 17 differs from existing results by directly comparing the ℓ_q risk of estimators with the minimax risk, instead of finding upper bounds for the ℓ_q loss. It is based on the oracle inequality for $\lambda>\lambda_{univ}$ in Theorem 10. However, in practice, a penalty level $\lambda<\lambda_{univ}$ is often empirically the best choice. As we mentioned in Remark 2, $\lambda_{mm}<\lambda_{univ}$ iff $R>\sigma/\sqrt{n}$. For $\lambda_{mm}/\lambda_{univ}=o(1)$,

oracle inequalities requiring penalty levels $\lambda \geq \lambda_{univ}$ do not match the order of the minimax lower bounds in Theorem 1. For example, when $p = n\log n$ and $d \asymp R^r/\lambda_{mm}^r \asymp n/\log\log n$, $\lambda_{mm}/\lambda_{univ} \to 0$ as $n \to \infty$ and the regularity conditions on X may still hold. Theorem 19 below closes this gap by providing the rate minimaxity of the Lasso in the quantiles of the ℓ_q loss with $\lambda \asymp \lambda_{mm} = o(\lambda_{univ})$. Define

$$CIF^*_{q,relax}(\xi,k,d) = \inf_{|A|=d}\Big\{\frac{d^{1/q}\|\Sigma_{A,*}u\|}{d^{1/2}\|u\|_q} : \min_{|A\setminus J|=d-k}\|u_{J^c}\|_1 < \xi d^{1/2}\|u_A\|\Big\}, \tag{49}$$

$$M^*_{q,relax}(\lambda,\rho,k,d) = \sup_{|A|=d}\Big\{\|u\|_q : \|\Sigma_{A,*}u\| \leq d^{1/2}\lambda, \min_{|A\setminus J|=d-k}\|u_{J^c}\|_1 \leq \rho\Big\}. \tag{50}$$

Theorem 19 *Let* $\lambda = \min\big(\lambda_{univ}, (1+\varepsilon_0)(1+\delta_d^+)^{1/2}\lambda_{mm}\big)/\alpha$ *with* $0 < \varepsilon_0 \leq \alpha < 1$ *and* $\{\lambda_{mm}, \lambda_{univ}, \delta_d^+\}$ *in (5) and (22). Let* $0 < r \leq 1 \leq q \leq 2$. *Suppose* $n \wedge p \geq d \asymp R^r/\lambda_{mm}^r \to \infty$, $\delta_d^+ = O(1)$ *and* $\lambda_{mm}n^{1/2}/\sigma \to \infty$. *Suppose that for certain* $k+\ell = d$ *with* $k \asymp \ell$,

$$\max\Big\{1/CIF^*_{q,relax}(\xi,k,d), M^*_{q,relax}(d^{-1/2}, d^{1/2}, k, d)\Big\} = O(1).$$

Then, the Lasso is rate minimax in ℓ_r *balls in the sense that for all* $\varepsilon > 0$,

$$\begin{aligned} &\inf\Big[t : \sup_{\|\beta\|_r \leq R} P_{\beta,X}\Big\{\|\widehat{\beta}^{(L)} - \beta\|_q^q \geq t^q R^r \lambda_{mm}^{q-r}\Big\} \leq \varepsilon\Big] \\ \lesssim\ &\inf\Big[t : \inf_{\delta}\sup_{\|\beta\|_r \leq R} P_{\beta,X}\Big\{\|\delta(X,y) - \beta\|_q^q \geq t^q R^r \lambda_{mm}^{q-r}\Big\} \leq \varepsilon\Big] < \infty. \end{aligned} \tag{51}$$

In particular, (51) holds if $1/(1-\delta_d^-) + \delta_d^+ = O(1)$ *as in Remark 18.*

The quantities (8), (12), (35), (40), (41), (49) and (50) are best understood by comparisons with functions of (22) and (43) via Propositions 5 and 15. These quantities also facilitate comparisons between our and existing upper bounds on the loss as in the derivation of Corollary 7. In such comparisons, the Hölder inequality and (22) give

$$\eta_{q,d} \leq \eta^*_{q,d},\ \ \kappa_{q,d,\ell} \leq \kappa^*_{q,d,\ell},\ \ \eta^*_{2,d} = 1/(1-\delta_d^-),\ \ \kappa^*_{2,d,\ell} \leq \theta_{d,\ell}\eta^*_{2,d},$$

where $\kappa^*_{q,d,\ell} = a_q^{1-1/q}\max_{A,B}\|\Sigma_A^{-1}\Sigma_{A,B}\|_{q,q}$ with $|A| = d$, $|B| = \ell$ and $A \cap B = \emptyset$.

5. Discussion

Although this paper focuses on the estimation of regression coefficients, the estimation of $X\beta^*$ (prediction) is an important problem (Greenshtein and Rotiv, 2004). Similar to the proof of (11), (9), (14) and (15) imply

$$\|X\widehat{\beta}^{(L)} - X\beta^*\|^2/n + 2\lambda\|(\widehat{\beta} - \beta^*)_J\|_1/(\xi+1) \leq \frac{\{2\xi/(\xi+1)\}|J|\lambda^2}{SCIF_{1,0}(\xi,J)} \tag{52}$$

in the event $z^*_\infty = \|X'(y - X\beta^*)/n\|_\infty \leq \lambda(\xi-1)/(\xi+1)$. Since $SCIF_{1,0}(\xi,J) \geq RE_{1,0}(\xi,J)$, (52) implies Lemma 2.1 of van de Geer and Bühlmann (2009) for $(z^*_\infty, \xi) = (0,1)$.

As we have explained in Section 3.2, the use of $\|\Sigma u\|_\infty$ in the numerator of (8) and (9) seems necessary to ensure the dominance of Theorem 3 over the oracle inequalities of the type (30). However, if we make the numerator quadratic in u, Corollary 7 still holds up to a constant factor with the following weak CIF:

$$CIF^w_{q,\ell}(\xi,J) = \inf_{u\in\mathscr{C}(\xi,J)} \frac{|J|^{1/q}u'_A\Sigma_{A,*}u}{\|u_J\|_1\|u_A\|_q},\ SCIF^w_{q,\ell}(\xi,J) = \inf_{u\in\mathscr{C}_-(\xi,J)} \frac{|J|^{1/q}u'_A\Sigma_{A,*}u}{\|u_J\|_1\|u_A\|_q}, \tag{53}$$

where $A = \arg\max_{|A\setminus J|\le\ell}\|u_A\|$. For example, (26), (27) and (28) are still consequences of Theorem 3 when (53) is used instead of (8) and (9).

Since the oracle inequalities in this paper apply directly to data points (X,y) and target vectors β^*, the normality assumption on the error in (3) is not crucial for the upper bounds for the estimation risk and loss (not even the condition $E_{\beta,X}y = X\beta$). For example, for the estimation of a target β^* with $X\beta^* \approx Ey$, the upper bounds in Theorem 19 are valid for $\|\widehat{\beta}^{(L)} - \beta^*\|_q^q$ with large probability under P and $\sigma = \sigma_1 + \sigma_2$, provided that

$$\begin{cases} E\exp(v'X'(y-Ey)) \le \exp(-n\sigma_1^2 v'\Sigma v/2), \\ \max_{|A|=\ell}\|P_A(Ey - X\beta^*)\| \le \sigma_2\sqrt{2\ell\log(p/\ell)}. \end{cases}$$

For design matrices X with iid sub-Gaussian rows, our results can be extended to β in ℓ_r balls with $1 < r \le 2$ due to $\sigma_2 \le O(\|\beta_{J^c}\|)$ for $\beta^*_J = \beta_J$ and $\beta^*_{J^c} = 0$.

The proofs in this paper do not completely deal with the most difficult case of $q > 2$ and $\lambda_{mm} = o(\lambda_{univ})$. For example, an extension of Theorem 12 to $q > 2$ seems to require sharp upper bounds for $z^*_{q,d}$ in (38).

For $\lambda < \lambda_{univ}$, the proof of Theorem 12 can be extended to the Dantzig selector with the feasibility of β^* replaced by the feasibility of $\widehat{\beta}^{(L)}$. This would yield slightly worse error bounds than those in Theorem 12. However, if we modify the Dantzig selector as

$$\widetilde{\beta} = \arg\min_b \left\{\|b\|_1 : \max_{|A|=d}\|X'_A(y-Xb)\| \le \sqrt{d}\lambda\right\}, \tag{54}$$

the feasibility of β^* would be guaranteed in the event $z^*_{2,d} \le \lambda$ even for $\lambda = o(\lambda_{univ})$ as in Theorems 12 and 19. This will provide sharper error bounds for the smaller λ and $q \le 2$. We omit this modification since the computational issues with (54) is not clear for $d > 1$.

Acknowledgments

This project is partially supported by the National Science Foundation (NSF grants DMS-0604571, DMS-0804626, DMS-0906420) and the National Security Agency (NSA grant H98230-09-1-0006).

Appendix A. Proofs

We provide all the proofs here. Lemms are stated and proved as needed.

Proof of Theorem 1. Let $\Theta = \Theta_{r,R}$ and $k = R^r/\lambda_{mm}^r$ for $r > 0$ and $\Theta = \Theta_{0,k}$ for $r = 0$. By (5), $\lambda_{mm}/\sigma_n = \sqrt{2\log(p/k) - r\log(\lambda_{mm}/\sigma_n)}$ for $r > 0$. Since $\lambda_{mm}/\sigma_n \to \infty$,

$$\lambda_{mm} = (1+o(1))\sigma_n\sqrt{2\log(p/k)},\ \min(k,p/k) \to \infty, \quad \forall\, r \geq 0. \tag{55}$$

Let $P_{\mu,w}$ be a (prior) probability distribution under which (z_j, β_j) are iid vectors with

$$z_j|\beta_j \sim N(\beta_j, \sigma_n^2),\ P_{\mu,w}\{\beta_j = \mu\} = w = 1 - P_{\mu,w}\{\beta_j = 0\},$$

where $\mu = \lambda_{mm}(1-\varepsilon)$ and $w = (1-\varepsilon)k/p$. Since $\widetilde{z}_j = x_j'(y - \sum_{i\neq j}\beta_i x_i)/n$ is sufficient for β_j given $(X, y, \beta_i, i \neq j)$ and $\widetilde{z}_j|\beta \sim z_j|\beta$, the minimum Bayes risk is bounded by

$$\begin{aligned}
\inf_{\widehat{\beta}} E_{\mu,w}E_{\beta,X}\|\widehat{\beta} - \beta\|_q^q &\geq E_{\mu,w}\sum_{j=1}^p \min_t E_{\beta,X}\Big[|t-\beta_j|^q\Big|X,y\Big] \\
&\geq E_{\mu,w}\sum_{j=1}^p \min_t E_{\beta,X}\Big[|t-\beta_j|^q\Big|X,y,\beta_i,i\neq j\Big] \\
&= E_{\mu,w}\sum_{j=1}^p \min_t E_{\beta,X}\Big[|t-\beta_j|^q\Big|z_j\Big] \\
&= (1+o(1))k\lambda_{mm}^q
\end{aligned} \tag{56}$$

as $k \wedge (p/k) \to \infty$ and then $\varepsilon \to 0$. The last step above is by Donoho and Johnstone (1994).

Let $N = \#\{j : \beta_j \neq 0\}$. Under $P_{\mu,w}$, $\|\beta\|_r^r = N\mu^r$ and $\|\beta\|_q^q = N\mu^q$, so that

$$\beta \in \Theta \ \Leftrightarrow\ N \leq k/(1-\varepsilon)^r \ \Rightarrow\ \|\beta\|_q^q \leq k\lambda_{mm}^q,\ \forall\, r \geq 0, \tag{57}$$

due to $R^r/\mu^r = k/(1-\varepsilon)^r$ and $\{k/(1-\varepsilon)^r\}\mu^p/\lambda_{mm}^p = k(1-\varepsilon)^{q-r} \leq k$ for $r > 0$. Let

$$\delta^* = \arg\min_\delta E_{\mu,w}E_{\beta,X}\big[\|\delta(X,y) - \beta\|_q^q\big|X,y,\beta\in\Theta\big].$$

Since the conditional Bayes risk of δ^* is no greater than the minimax risk in Θ,

$$\begin{aligned}
& (1+o(1))k\lambda_{mm}^q \\
\leq\ & E_{\mu,w}E_{\beta,X}\big[\|\delta^*-\beta\|_q^q\big|\beta\in\Theta\big] + E_{\mu,w}E_{\beta,X}\|\delta^*-\beta\|_q^q I\{\beta\notin\Theta\} \\
\leq\ & \mathscr{R}(\Theta;X) + 2^{(q-1)_+}E_{\mu,w}E_{\beta,X}\big(\|\delta^*\|_q^q + \|\beta\|_q^q\big)I\{\beta\notin\Theta\}.
\end{aligned} \tag{58}$$

Since $E_{\mu,w}E_{\beta,X}[\|\delta^*-\beta\|_q^q|X,y,\beta\in\Theta] \leq E_{\mu,w}[\|\beta\|_q^q|X,y,\beta\in\Theta] \leq k\lambda_{mm}^q$ by (57), $\|\delta^*\|_q^q \leq 2^{(q-1)_+}k\lambda_{mm}^q$ a.s. Thus, since $N \sim \text{Binomial}(p,w)$ with $pw = (1-\varepsilon)k \to \infty$,

$$\begin{aligned}
& E_{\mu,w}E_{\beta,X}\big(\|\delta^*\|_q^q + \|\beta\|_q^q\big)I\{\beta\notin\Theta\} \\
\leq\ & 2^{(q-1)_+}k\lambda_{mm}^q P_{\mu,w}\{N > wp/(1-\varepsilon)\} + \mu^q E_{\mu,w}NI\{N > wp/(1-\varepsilon)\} \\
=\ & o(1)k\lambda_{mm}^q
\end{aligned} \tag{59}$$

by (57). The combination of (58) and (59) gives (6).

Now consider the loss $L(\delta,\beta) = I\{\|\delta-\beta\|_q > ck^{1/q}\lambda_{mm}\}$ in (7). Define

$$\widehat{\beta} = \delta(X,y)I\Big\{\|\delta(X,y)\|_q \leq (1+c)k^{1/q}\lambda_{mm}\Big\}.$$

By (57), $\beta \in \Theta$ implies $\|\beta\|_q^q \le k\lambda_{mm}^q$ and

$$\begin{aligned}\|\widehat{\beta}-\beta\|_q^q &\le c^q k\lambda_{mm}^q I\Big\{\|\delta-\beta\|_q \le ck^{1/q}\lambda_{mm}\Big\} \\ &\quad +\Big(\|\beta\|_q+(1+c)k^{1/q}\lambda_{mm}\Big)^q I\Big\{\|\delta-\beta\|_q > ck^{1/q}\lambda_{mm}\Big\}.\end{aligned}$$

Since $\|\beta\|_q^q = N\mu^q \le N\lambda_{mm}^q$, it follows that

$$\begin{aligned}E_{\mu,w}E_{\beta,X}\|\widehat{\beta}-\beta\|_q^q &\le c^q k\lambda_{mm}^q + (2+c)^q k\lambda_{mm}^q \max_{\beta\in\Theta} E_{\beta,X}L(\delta(X,y),\beta) \\ &\quad +2^{q-1}\lambda_{mm}^q E_{\mu,w}\Big(N+(1+c)^q k\Big)I\{\beta\notin\Theta\}. \qquad (60)\end{aligned}$$

Since $E_{\mu,w}\Big(N+(1+c)^q k\Big)I\{\beta\notin\Theta\} = o(1)k$ by (57), (56) and (60) yield

$$\sup_{\beta\in\Theta} E_{\beta,X}L(\delta(X,y),\beta) \ge \frac{1-c^q+o(1)}{(2+c)^q}.$$

Since the $o(1)$ is uniform in the choice of $\delta(X,y)$, we find

$$\inf_{\delta}\sup_{\beta\in\Theta} P_{\beta,X}\Big\{\|\delta(X,y)-\beta\|_q^q > (1-\varepsilon)k\lambda_{mm}^q\Big\} \ge \frac{\varepsilon+o(1)}{3^q},\ \forall 0<\varepsilon<1.$$

This gives (7) and completes the proof. ■

Proof of Theorem 3. Part (i) follows from (14) and (15) as briefly explained in the paragraph below the statement of the theorem. For the Dantzig selector, $z^* \le \lambda$ implies (15) and the feasibility of β^* for the ℓ_∞ constraint in (2), and the feasibility of β^* implies (14). For the Lasso estimator, (14) and (15) follow from the Karush-Kuhn-Tucker conditions

$$\|x_j(y-X\widehat{\beta})/n\|_\infty \le \lambda, \quad \widehat{\beta}_j \ne 0 \ \Rightarrow\ x_j(y-X\widehat{\beta})/n = \mathrm{sgn}(\widehat{\beta}_j)\lambda.$$

Part (ii) follows from $P_{\beta,X}\{z_\infty^* > t\sigma/\sqrt{n}\} \le 2pP\{N(0,1)>t\}$. ■

In this paper and those cited in Section 3.2, tails of ℓ_q norms or inner products are bounded by shifting inequalities (Candes and Tao, 2007, Lemma 3.1). The following lemma combines and extends the sharp shifting inequalities of Cai, Wang, and Xu (2010) for $q=2$ and Ye and Zhang (2009) for $w'h$ with $q=\infty$.

Lemma 20 *Let $1 \le q \le \infty$ and $a_q = (1-1/q)/q^{1/(q-1)}$ with $a_\infty = 1$. Let $h \in \mathbb{R}^p$, $J \subset \{1,\ldots,p\}$ and A be the union of J and the indices of the ℓ largest $|h_j|$ with $j \notin J$, $1 \le \ell \le p-|J|$. Then, $\|h_{A^c}\|_q \le (a_q/\ell)^{1-1/q}\|h_{J^c}\|_1$. Moreover, for any vector $w \in \mathbb{R}^p$,*

$$\sum_{j\notin A} w_j h_j \le \|h_{J^c}\|_1\Big(\frac{a}{\ell}\vee\frac{a_q}{\ell}\Big)^{1-1/q}\max\Big\{\|w_B\|_{q/(q-1)} : B\cap A=\emptyset, |B|\le\lceil\ell/a\rceil\Big\}. \qquad (61)$$

Proof. We first prove that for all decreasing functions $h(t)\ge 0$,

$$\sum_{m=0}^{\infty}\Big(\int_{\ell+m\ell/a}^{\ell+(m+1)\ell/a} h^q(t)dt\Big)^{1/q} \le \max\big\{1,(a_q/a)^{1-1/q}\big\}\frac{a^{1-1/q}}{\ell^{1-1/q}}\int_0^\infty h(t)dt. \qquad (62)$$

With $x = at/\ell - m$ and possibly different $h \downarrow 0$, the above inequality is a consequence of

$$\max\Big\{\int_a^{1+a} h^q(x)dx : \int_0^1 h(x)dx = 1\Big\} \le \max\big\{1, (a_q/a)^{q-1}\big\} \tag{63}$$

It suffices to consider $0 < a \le a_q$. Since $\int_a^{1+a} h^q(x)dx$ is convex in h, it suffices to consider $h(x) = v + (u-v)I\{x \le w\}$ for certain $u \ge 1 \ge v$ and $a \le w \le 1$. Since $\int_0^1 h(x)dx = 1$, $v = (1-uw)/(1-w)$. Thus, for fixed w, $\int_a^{1+a} h^q(x)dx$ is convex in u, and its maximum is attained at the extreme points $u \in \{1, 1/w\}$. For $u = 1/w$, we have $v = 0$ and $\int_a^{1+a} h^q(x)dx = u^{q-1} - au^q$, so that the optimal u satisfies $au = (q-1)/q$, resulting in the maximum $\{(q-1)/(qa)\}^{q-1}/q = (a_q/a)^{q-1}$. This yields (63) since $\int_a^{1+a} h^q(x)dx = 1$ at the other extreme $u = 1$. Thus, $\|h_{A^c}\|_q \le (a_q/\ell)^{1-1/q}\|h_{J^c}\|_1$ by (62), and (61) follows with an application of the Hölder inequality to $\int_{\ell+m\ell/a}^{\ell+(m+1)\ell/a} w(t)h(t)dt$. ■

Proof of Proposition 5. Let $k = |J|$. Since $\|u_{A^c}\|_q \le (a_q/\ell)^{1-1/q}\|u_{J^c}\|_1$ by Lemma 20 and $\|u_{J^c}\|_1 \le \xi\|u_J\|_1 \le \xi k^{1-1/q}\|u_A\|_q$, $\|u\|_q \le C_{q,q}(\xi, k/\ell)\|u_A\|_q$. By the Hölder inequality $\|u\|_s \le \|u\|_1^{(1/s-1/q)/(1-1/q)}\|u\|_q^{(1-1/s)/(1-1/q)} = \|u\|_1^{1/s_1}\|u\|_q^{1-1/s_1}$ with $s_1 = (1-1/q)/(1/s-1/q) \ge s$. Since $\|u\|_1 \le (1+\xi)\|u_J\|_1 \le (1+\xi)k^{1-1/q}\|u_A\|_q$,

$$\|u\|_s k^{1/q-1/s}/\|u_A\|_q \le (1+\xi)^{1/s_1} C_{q,q}^{1-1/s_1}(\xi, k/\ell) = C_{s,q}(\xi, k/\ell).$$

This gives (17). Since $v'_{J^c}\Sigma_{J^c,*}u \le 0$ for $u \in \mathscr{C}_-(\xi, J)$ and $v \in \mathscr{Q}(u, J)$, the first parts of (18) and (19) follow from (8) and (9). For $h = u$ and the choice of A in Lemma 20, an application of (61) with $w = (v'_A\Sigma_{A,*})'$ yields

$$\begin{aligned} v'_A\Sigma_{A,*}u &= v'_A\Sigma_A u_A + v'_A\Sigma_{A,A^c}u_{A^c} \\ &\ge v'_A\Sigma_A u_A - \max_B \|\Sigma_{B,A}v_A\|_{r/(r-1)}(a_r/\ell)^{1-1/r}\|u_{J^c}\|_1. \end{aligned} \tag{64}$$

Since $\|u_{J^c}\|_1 \le \xi\|u_J\|_1 \le \xi k^{1-1/q}\|u_A\|_q$, (64) and (16) imply

$$\begin{aligned} \frac{v'_A\Sigma_{A,*}u}{\|v_A\|_1\|u_A\|_q/k^{1/q}} &\ge \frac{v'_A\Sigma_A u_A - \xi\|u_A\|_q k^{1-1/q}(a_r/\ell)^{1-1/r}\max_B\|\Sigma_{B,A}v_A\|_{r/(r-1)}}{\|v_A\|_1\|u_A\|_q/k^{1/q}} \\ &= \phi_{q,\ell}(u, A, r, w, B; \xi, J)k^{1/q}/\|w_A\|_1 \end{aligned}$$

with $w = v\|u_A\|_q/(v'_A\Sigma_A u_A)$. This gives the second parts of (18) and (19). ■

Proof of Theorem 8. This theorem is a direct consequence of (10) with $q = \infty$, since $\|\widehat{\beta} - \beta\|_\infty \le \lambda' < \min_{\beta_j \ne 0}|\beta_j|/2$ guarantees $\{j : |\widehat{\beta}_j| > \lambda'\} = \{j : \beta_j \ne 0\}$. ■

Proof of Theorem 10. Let $h = \widehat{\beta} - \beta^*$ for either estimator. As in (14) and (15),

$$\|\Sigma h\|_\infty \le \xi_1\lambda,\ \|h_{J^c}\|_1 \le \xi_0\|h_J\|_1 + (\xi_0+1)\rho_J, \tag{65}$$

in the given events, with $\{\xi_0, \xi_1\} = (1, 2)$ for the Dantzig selector and the $\{\xi_0, \xi_1\}$ in (37) for the Lasso. It follows from Theorem 3 that (36) and (37) hold for $\|h_{J^c}\|_1 \le \xi\|h_J\|_1$, or equivalently $h \in \mathscr{C}(\xi, J)$. By (65), it remains to consider

$$\xi\|h_J\|_1 \le \|h_{J^c}\|_1 \le \xi_0\|h_J\|_1 + (\xi_0+1)\rho_J.$$

Since $\xi > \xi_0$, $\|h_J\|_1 \le (\xi_0+1)\rho_J/(\xi-\xi_0)$ in this case, so that $\|h\|_1 \le (1+\xi_0)\|h_J\|_1 + (\xi_0+1)\rho_J \le \rho_J(1+\xi_0)(1+\xi)/(\xi-\xi_0) = \xi_2\rho_J$. Thus, (35) gives $\|h\|_q \le M_q(\xi_1\lambda, \xi_2\rho_J)$. ■

Proof of Theorem 12. Let $h = \widehat{\beta}^{(L)} - \beta^*$, $z = X'(y - X\beta^*)/n$, $k = |J|$, $\ell = d - k$, and $A = \arg\max_{|B|=d, B\supset J} \|h_B\|$ as in (40). Since $k \le 3d/4$, $4\ell \ge d$. Lemma 20 with $\{w, q, a_q\} = \{z, 2, 1/4\}$ yields $h'_{A^c} z_{A^c} \le \|h_{J^c}\|_1 z^*_{2,4\ell} \le \|h_{J^c}\|_1 z^*_{2,d}$, so that

$$\begin{aligned} \|Xh\|^2/n &= h'_A z_A + h'_{A^c} z_{A^c} - h'X'(y - X\widehat{\beta}^{(L)})/n \\ &\le \sqrt{d}\|h_A\| z^*_{2,d} + \|h_{J^c}\|_1 z^*_{2,d} - \lambda\|\widehat{\beta}^{(L)}\|_1 + \lambda\|\beta^*\|_1. \end{aligned}$$

Since $-\lambda\|\widehat{\beta}^{(L)}\|_1 + \lambda\|\beta^*\|_1 \le -\lambda\|h_{J^c}\|_1 + \lambda\|h_J\|_1 + 2\lambda\rho_J$ and $\|h_J\|_1 \le \sqrt{3d/4}\|h_A\|$,

$$\|Xh\|^2/n + (\lambda - z^*_{2,d})\|h_{J^c}\|_1 \le (\lambda + z^*_{2,d})\sqrt{d}\|h_A\| + 2\lambda\rho_J.$$

Since $\Sigma_{A,*}h = z_A - X'_A(y - X\widehat{\beta}^{(L)})/n$, $\|\Sigma_{A,*}h\| \le (z^*_{2,d} + \lambda)\sqrt{d}$. Thus, as in (65),

$$\|\Sigma_{A,*}h\| \le \xi_1\sqrt{d}\lambda,\ \|h_{J^c}\|_1 \le \xi_0\sqrt{d}\|h_A\| + (\xi_0 + 1)\rho_J. \tag{66}$$

For $\|h_{J^c}\|_1 \le \xi\sqrt{d}\|h_A\|$, (42) follows from

$$d^{1/2-1/q}\|h\|_q CIF_{q,relax}(\xi, J, d) \le \|\Sigma_{A,*}h\| \le \xi_1\sqrt{d}\lambda.$$

For $\xi\sqrt{d}\|h_A\| \le \|h_{J^c}\|_1$, the second inequality of (66) gives $\sqrt{d}\|h_A\| \le (\xi_0 + 1)\rho_J/(\xi - \xi_0)$ and then $\|h\|_1 \le (1 + \xi_0)(\sqrt{d}\|h_A\| + \rho_J) \le \xi_2\rho_J$. Thus, $\|h\|_q \le M_{q,relax}(\xi_1\lambda, \xi_2\rho_J, J, d)$ by (41), in view of the first inequality of (66). ■

Proof of Proposition 15. Let A be the index set of the k largest u_j and w satisfy $\|\Sigma_A w_A\|_{q/(q-1)} = w'_A\Sigma_A u_A/\|u_A\|_q = 1$. By Lemma 20, $w'_A\Sigma_{A,A^c}u_{A^c} \le \kappa_{q,k,k}k^{1/q-1}\rho$, so that $\|u_A\|_q = w'_A\Sigma_A u_A \le \|w_A\|_1\|\Sigma u\|_\infty + \kappa_{q,k,k}k^{1/q-1}\rho \le \eta_{q,k}k^{1/q}\lambda + \kappa_{q,k,k}k^{1/q-1}\rho$. This and $\|u\|_q \le (\|u_A\|_q^q + (a_q/k)^{q-1}\rho^q)^{1/q}$ from Lemma 20 yields (44).

Let A be as in (41) and $w_A = \Sigma_A^{-1}u_A/\|u_A\|$. For $\|\Sigma_{A,*}u\| \le \sqrt{d}\lambda$ and $\|u\|_1 \le \rho$, $\|u_A\| = w'_A\Sigma_A u_A \le \|w_A\|d^{1/2}\lambda + w_A\Sigma_{A,A^c}u_{A^c} \le \eta^*_{2,d}d^{1/2}\lambda + \kappa_{2,d,\ell}\ell^{-1/2}\rho$, so that (45) follows from $\|u\|_q^q \le (d^{1/q-1/2}\|u_A\|)^q + (a_q/\ell)^{q-1}\rho^q$.

Let u and A be as in (40) and $k = |J|$. Similar to the proof of Proposition 5, we have $\|u\|^2 \le \{1 + \xi^2 k/(4\ell)\}\|u_A\|^2$ and $\|u\|_1 \le \sqrt{k}\|u_A\| + \|u_{J^c}\|_1 \le (1 + \xi\sqrt{d/k})k^{1/2}\|u_A\|$. Thus, for $1 \le q \le 2$, $\|u\|_q \le \|u\|_1^{2/q-1}\|u\|^{2-2/q} \le C_{q,2}(\xi(d/k)^{1/2}, k^2/(d\ell))k^{1/q-1/2}\|u_A\|$. Since $\|u_{J^c}\|_1 \le \xi d^{1/2}\|u_A\|$ and (64) holds for $v = u$, $u'_A\Sigma_{A,*}u \ge \phi_{2,\ell}(u, A, r, w, B; \xi', J)/\|w_A\|$ for $\|u_A\| = 1$, where $\xi' = \xi\sqrt{d/k}$ and $w = u/(u'_A\Sigma_A u_A)$. Thus, (20) gives (46). ■

The proof of Theorem 17 requires the following lemma.

Lemma 21 *Let $\widehat{\beta}$ be either the Dantzig or the Lasso estimator at penalty level λ. Suppose $\|\beta\|_r \le R$ with $0 < r \vee 1 \le q$. For any event Ω_0 with $t_* = \sqrt{2\log(1/P_{\beta,X}(\Omega_0))} \ge 1$,*

$$E_{\beta,X}\|\widehat{\beta} - \beta\|_q^q I_{\Omega_0} \le 2^{q-1}P_{\beta,X}(\Omega_0)\Big\{\frac{\Gamma(2q+1)}{(t_*^2 n\lambda/\sigma^2)^q} + \Big(\frac{(t_* + \sqrt{n})^2}{n\lambda/\sigma^2} + 2p^{(1-1/r)_+}R\Big)^q\Big\}. \tag{67}$$

In particular, if $(\log p)/n + \sigma^2/(n\lambda^2) + R^r/(n\lambda^r) + \lambda^r/R^r = O(1)$, then

$$E_{\beta,X}\|\widehat{\beta} - \beta\|_q^q I_{\Omega_0} = o(1)R^r\lambda^{q-r}, \tag{68}$$

provided that $P_{\beta,X}(\Omega_0)(\lambda^r/R^r)\{(\sigma/\lambda)^{2q} + p^{q(1-1/r)_+}(R/\lambda)^q\} = o(1)$.

Remark 22 *Since the unit sphere $S^{n-1} \subset \mathbb{R}^n$ is covered by $(2/\varepsilon+1)^n$ ε-balls for all $\varepsilon > 0$, a certain ε ball contains at least m unit vectors $x_j/\|x_j\|$ for $\varepsilon = (\log(p/m))/(2n)$. It follows that the set of design vectors x_j contains some highly correlated clusters when $(\log p)/n \geq 2$. Thus, the condition $(\log p)/n = O(1)$ is natural for the estimation of β.*

Proof. Let $\widehat{\beta}$ be the Lasso estimator. Since $\widehat{\beta}$ minimizes the penalized loss, $\lambda\|\widehat{\beta}\|_1 \leq \|\varepsilon\|^2/(2n) + \lambda\|\beta\|_1$, so that

$$\|\widehat{\beta}\|_1 + \|\beta\|_1 \leq \frac{\|\varepsilon\|^2}{2n\lambda} + 2\|\beta\|_1 \leq \frac{(\|\varepsilon\|/\sigma - t_* - \sqrt{n})_+^2}{n\lambda/\sigma^2} + \frac{(t_* + \sqrt{n})^2}{n\lambda/\sigma^2} + 2p^{(1-1/r)_+}R.$$

Since $\|\varepsilon/\sigma\|$ is a Lipschitz(1) function of $\varepsilon/\sigma \sim N(0, I_n)$ and $E_{\beta,X}\|\varepsilon\|/\sigma \leq \sqrt{n}$, the Gaussian isoperimetric theorem gives $P_{\beta,X}\{\|\varepsilon\|/\sigma - \sqrt{n} > t\} \leq e^{-t^2/2}$, so that

$$\begin{aligned} E_{\beta,X}(\|\varepsilon\|/\sigma - t_* - \sqrt{n})_+^{2q} &\leq \int_0^\infty P_{\beta,X}\{\|\varepsilon\|/\sigma - t_* - \sqrt{n} > t\} dt^{2q} \\ &\leq \int_0^\infty e^{-t_*^2/2 - t_* t} dt^{2q} = P_{\beta,X}(\Omega_0)\Gamma(2q+1)/t_*^{2q}. \end{aligned}$$

The above inequalities yield (67) due to $\|\widehat{\beta} - \beta\|_q^q \leq (\|\widehat{\beta}\|_1 + \|\beta\|_1)^q$ for $q \geq 1$.

It follows from (67) that

$$\begin{aligned} & E_{\beta,X}\|\widehat{\beta} - \beta\|_q^q I_{\Omega_0}/\{R^r\lambda^{q-r}\} \\ = \;& O(\lambda^r/R^r)P_{\beta,X}(\Omega_0)\Big\{O(1) + (t_*^2/n+1)^q(\sigma/\lambda)^{2q} + p^{q(1-1/r)_+}R^q/\lambda^q\Big\}. \end{aligned}$$

Since the right-hand side is of no greater order than $P_{\beta,X}(\Omega_0)\{(t_*^2+n)^q + p^q n^{q/r}\} = o(1)$ for $t_*^2/(n \vee \log p) \to \infty$, it suffices to consider the case $t_*^2/n = O(1)$. Hence, (68) holds under the specified conditions. The same conclusions hold for the Dantzig selector, since $\|\widehat{\beta}^{(D)}\|_1 \leq \|\widehat{\beta}^{(L)}\|_1$. ■

Proof of Theorem 17. We first bound $\lambda_{univ}/\lambda_{mm}$ and the expected loss for large $z_\infty^* = \|X'\varepsilon/n\|_\infty$. Let $\sigma_n = \sigma/\sqrt{n}$. Since $R^r/\lambda_{mm}^r \asymp d$, (5) and (55) give

$$2\sigma_n^2 \log(p/d) \approx \lambda_{mm}^2. \tag{69}$$

Since $(\log d)/\log p \leq c_0$, $(1-c_0)\lambda_{univ}^2 = (1-c_0)2\sigma_n^2 \log p \leq 2\sigma_n^2 \log(p/d) \approx \lambda_{mm}^2$ and

$$C_1 \approx C_1^* = 2(\lambda/\lambda_{mm})(d\lambda_{mm}^r/R^r)^{1/q},\ C_2 \approx C_2^* = R/(\lambda d^{1/r}).$$

Let $\Omega_0 = \{z_\infty^*/\lambda > \alpha_1\}$. Since z_∞^* is the maximum of p variables from $N(0, \sigma_n^2)$, $P_{\beta,X}\{\Omega_0\} \ll p\exp(-n(\alpha_1\lambda)^2/(2\sigma^2)) \leq p^{1-(\alpha_1/\alpha_0)^2}$ for large n. Thus, due to $\lambda^2/\sigma_n^2 \asymp \log p$ and $n \geq d \asymp R^r/\lambda_{mm}^r \asymp R^r/\lambda^r \to \infty$, we have

$$P_{\beta,X}(\Omega_0)(\lambda^r/R^r)\{(\sigma/\lambda)^{2q} + (R/\lambda)^q\} = o(1)p^{1-(\alpha_1/\alpha_0)^2}(n^q/d + d^{q/r-1}) = o(1).$$

Since $0 < r \leq 1$, Lemma 21 gives $E_{\beta,X}\|h\|_q^q I\{z_\infty^*/\lambda > \alpha_1\} = o(R^r\lambda_{mm}^{q-r})$, where $h = \widehat{\beta} - \beta$.

Next we prove $E_{\beta,X}\|h\|_q^q I\{z_\infty^*/\lambda \leq \alpha_1\} = O(R^r\lambda_{mm}^{q-r})$. Consider $z_\infty^* \leq \alpha_1\lambda$. By (65) with $J = \emptyset$, $\|h\|_q \leq 2M_q(\lambda, \xi'\|\beta\|_1)$ with $\xi' = \xi^*$ for $r = 1$. Since $M_q(\lambda, \rho) = cM_q(\lambda/c, \rho/c)$,

$$\|h\|_q/(R^{r/q}\lambda_{mm}^{1-r/q}) \leq C_1^* M_q(d^{-1/q}, \xi_2' C_2^* d^{1/r-1/q}).$$

This gives (47) for $r = 1$ and $E_{\beta,X}\|h\|_q^q I\{\alpha\lambda < z_\infty^* \le \alpha_1\lambda\} = o(p^{1-(\alpha/\alpha_0)^2} d^{q(1/r-1)}) = o(1)$ for $r < 1$. Let $J = \arg\max_{|A|=d} \|\beta_A\|_1$. For $\beta \in \Theta_{r,R}$, $\|\beta_{J^c}\|_\infty \le (R^r/d)^{1/r}$, so that $\rho_J/(\lambda d) \le (R^r/d)^{(1-r)/r} R^r/(\lambda d) = C_2^*$. Thus, in the event $z_\infty^* \le \alpha\lambda$, Theorem 10 gives

$$\|h\|_q/(R^{r/q}\lambda_{mm}^{1-r/q}) \le \max\left\{C_1^*/CIF_q^*(\xi,d), C_1^* M_q(d^{-1/q}, \xi^* C_2^* d^{1-1/q})\right\},\ r < 1.$$

It remains to prove (48) under $\eta_{q,d} + \kappa_{q,d,d} = O(1)$. In fact, by (44) it suffices to prove $1/CIF_q^*(\xi,k) = O(1)$ for any $k + \ell = d$ with $k \asymp d$. This follows from (21), since $CIF_{q,p}(\xi,|J|) \gtrsim \{1 - \xi(k/\ell)^{1-1/q}\kappa_{q,d,d}\}/\eta_{q,d} > 0$ uniformly for $|J| = k$ and small k/ℓ. ■

The proof of Theorem 19 requires the following simpler version of Lemma 2 in Zhang (2010).

Lemma 23 *Let $\widetilde{p}_\ell$ be the positive number satisfying $2\log\widetilde{p}_\ell - 1 - \log(2\log\widetilde{p}_\ell) = (2/\ell)\log\binom{p}{\ell}$. Suppose $\varepsilon \sim N(\mathbf{0}, \sigma^2 I_n)$ under probability P. Then,*

$$P\Big\{\max_{|A|=\ell} \|P_A\varepsilon\| \ge \sigma\sqrt{2\ell\log\widetilde{p}_\ell}\Big\} \le \frac{1}{2\sqrt{\log\widetilde{p}_\ell}} \le \frac{1}{\sqrt{2}},$$

where $P_A = X_A(X_A'X_A)^{-1}X_A'$ is the projection to the linear span of $\{x_j, j \in A\}$.

Proof of Theorem 19. Let $\gamma_d = (1+\delta_d^+)^{1/2}$ There are two cases. We omit the proof in the case of $\lambda = \lambda_{univ}/\alpha$ since it is identical to the second half of the proof of Theorem 17. It remains to consider the case $\lambda < \lambda_{univ}/\alpha$, that is, $(1+\varepsilon_0)\gamma_d\lambda_{mm} < \lambda_{univ}$.

For $|A| = d$, $\|X_A'\varepsilon/n\| \le \gamma_d\|P_A\varepsilon\|/\sqrt{n}$, so that by (38) and Lemma 23

$$P_{\beta,X}\Big\{z_{2,d}^* \le \gamma_d\sigma\sqrt{(2/n)\log\widetilde{p}_d}\Big\} \ge 1 - 1/(2\sqrt{\log\widetilde{p}_d}) \to 1.$$

Since $R^r/\lambda_{mm}^r \asymp d$ and $\lambda_{mm}n^{1/2}/\sigma \to \infty$, $\lambda_{mm} = (1+o(1))\sigma\sqrt{(2/n)\log(p/d)}$ by (69). By Stirling, $\log\binom{p}{d} = (1+o(1))d\log(p/d)$ for $p/d \to \infty$, so that $\lambda_{mm} = (1+o(1))\sigma\sqrt{(2/n)\log\widetilde{p}_d}$. Thus, $\gamma_d\sigma\sqrt{(2/n)\log\widetilde{p}_d} \le \alpha\lambda$ and $P_{\beta,X}\{z_{2,d}^* \le \alpha\lambda\} \to 1$.

Consider the event $z_{2,d}^* \le \alpha\lambda$. Since $\xi > (1+\alpha)/(1-\alpha)$, Theorem 12 asserts that for $J = \arg\max_{|A|=k}\|\beta_A\|_1$ and certain constants $\{\xi_1, \xi_2\}$,

$$\|h\|_q \le \max\Big\{\frac{\xi_1 d^{1/q}\lambda}{CIF_{q,relax}(\xi,J,d)}, M_{q,relax}\Big(\xi_1\lambda, \xi_2\rho_J, J, d\Big)\Big\}.$$

The rest of the proof is similar to the proof of Theorem 17 and omitted. ■

References

P. Bickel, Y. Ritov, and A. Tsybakov. Simultaneous analysis of lasso and dantzig selector. *Annals of Statistics*, 37:1705–1732, 2009.

F. Bunea, A. Tsybakov, and M. Wegkamp. Sparsity oracle inequalities for the lasso. *Electronic Journal of Statistics*, 1:169–194, 2007.

T. Cai, L. Wang, and G. Xu. Shifting inequality and recovery of sparse signals. *IEEE Transactions on Signal Processing*, 58:1300–1308, 2010.

E. Candes and Y. Plan. Near-ideal model selection by ℓ_1 minimization. *Annals of Statistics*, 37: 2145–2177, 2009.

E. Candes and T. Tao. The dantzig selector: statistical estimation when p is much larger than n (with discussion). *Annals of Statistics*, 35:2313–2404, 2007.

S. Chen and D. L. Donoho. On basis pursuit. Technical report, Department of Statistics, Stanford University, 1994.

D. L. Donoho and I. Johnstone. Minimax risk over ℓ_p–balls for ℓ_q–error. *Probability Theory and Related Fields*, 99:277–303, 1994.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression (with discussion). *Annals of Statistics*, 32:407–499, 2004.

B. Efron, T. Hastie, and R. Tibshirani. Discussion: The dantzig selector: statistical estimation when p is much larger than n. *Annals of Statistics*, 35:2358–2364, 2007.

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96:1348–1360, 2001.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156. Morgan Kauffmann, San Francisco, 1996.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion). *Annals of Statistics*, 28:337–407, 2000.

E. Greenshtein and Y. Rotiv. Persistence in high–dimensional linear predictor selection and the virtue of overparametrization. *Bernoulli*, 10:971–988, 2004.

J. Huang, S. Ma, and C.-H. Zhang. Adaptive lasso for sparse high–dimensional regression models. *Statistica Sinica*, 18:1603–1618, 2008.

V. Koltchinskii. The dantzig selector and sparsity oracle inequalities. *Bernoulli*, 15:799–828, 2009.

K. Lounici. Sup–norm convergence rate and sign concentration property of lasso and dantzig estimators. *Electronic Journal of Statistics*, 2:90–102, 2008.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34:1436–1462, 2006.

N. Meinshausen and B. Yu. Lasso–type recovery of sparse representations for high–dimensional data. *Annals of Statistics*, 37:246–270, 2009.

M. Osborne, B. Presnell, and B. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20:389–404, 2000a.

M. Osborne, B. Presnell, and B. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000b.

G. Raskutti, M. J. Wainwright, and B. Yu. Minimax rates of estimation for high–dimensional linear regression over ℓ_q–balls. Technical report, University of California, Berkeley, 2009.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 58:267–288, 1996.

J. A. Tropp. Just relax: convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52:1030–1051, 2006.

S. van de Geer. The deterministic lasso. Technical Report 140, ETH Zurich, Switzerland, 2007.

S. van de Geer. High–dimensional generalized linear models and the lasso. *Annals of Statistics*, 36: 614–645, 2008.

S. van de Geer and P. Bühlmann. On the conditions used to prove oracle results for the lasso. *Electronic Journal of Statistics*, 3:1360–1392, 2009.

M. J. Wainwright. Sharp thresholds for noisy and high–dimensional recovery of sparsity using ℓ_1–constrained quadratic programming (lasso). *IEEE Transactions on Information Theory*, 55: 2183–2202, 2009.

F. Ye and C.-H. Zhang. Rate minimaxity of the lasso and dantzig estimators. Technical report, Department of Statistics and Biostatistics, Rutgers University, 2009.

C.-H. Zhang. Least squares estimation and variable selection under minimax concave penalty. In *Mathematisches Forschungsintitut Oberwolfach: Sparse Recovery Problems in High Dimensions*, 3 2009a.

C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, 38:894–942, 2010.

C.-H. Zhang and J. Huang. The sparsity and bias of the lasso selection in high–dimensional linear regression. *Annals of Statistics*, 36:1567–1594, 2008.

T. Zhang. Some sharp performance bounds for least squares regression with l_1 regularization. *Annals of Statistics*, 37:2109–2144, 2009b.

P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2567, 2006.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

Journal of Machine Learning Research 11 (2010) 3541-3570 Submitted 4/10; Revised 10/10; Published 12/10

Incremental Sigmoid Belief Networks for Grammar Learning

James Henderson JAMES.HENDERSON@UNIGE.CH
Department of Computer Science
University of Geneva
7 route de Drize, Battelle bâtiment A
1227, Carouge, Switzerland

Ivan Titov TITOV@MMCI.UNI-SAARLAND.DE
MMCI Cluster of Excellence
Saarland University
Saarbrücken, Germany

Editor: Mark Johnson

Abstract

We propose a class of Bayesian networks appropriate for structured prediction problems where the Bayesian network's model structure is a function of the predicted output structure. These incremental sigmoid belief networks (ISBNs) make decoding possible because inference with partial output structures does not require summing over the unboundedly many compatible model structures, due to their directed edges and incrementally specified model structure. ISBNs are specifically targeted at challenging structured prediction problems such as natural language parsing, where learning the domain's complex statistical dependencies benefits from large numbers of latent variables. While exact inference in ISBNs with large numbers of latent variables is not tractable, we propose two efficient approximations. First, we demonstrate that a previous neural network parsing model can be viewed as a coarse mean-field approximation to inference with ISBNs. We then derive a more accurate but still tractable variational approximation, which proves effective in artificial experiments. We compare the effectiveness of these models on a benchmark natural language parsing task, where they achieve accuracy competitive with the state-of-the-art. The model which is a closer approximation to an ISBN has better parsing accuracy, suggesting that ISBNs are an appropriate abstract model of natural language grammar learning.

Keywords: Bayesian networks, dynamic Bayesian networks, grammar learning, natural language parsing, neural networks

1. Introduction

In recent years, there has been increasing interest in structured prediction problems, that is, classification problems with a large (or infinite) structured set of output categories. The set of output categories are structured in the sense that useful generalisations exist across categories, as usually reflected in a structured representation of the individual categories. For example, the output categories might be represented as arbitrarily long sequences of labels, reflecting generalisations across categories which share similar sets of sub-sequences. Often, given an input, the structure of the possible output categories can be uniquely determined by the structure of the input. For example in sequence labelling tasks, all possible output categories are label sequences of the same length as the input sequence to be labelled. In this article, we investigate structured classification problems

where this is not true; the structure of the possible output categories is not uniquely determined by the input to be classified. The most common type of such problems is when the input is a sequence and the output is a more complex structure, such as a tree. In reference to this case, we will refer to problems where the output structure is not uniquely determined by the input as "parsing problems". Such problems frequently arise in natural language processing (e.g., prediction of a phrase structure tree given a sentence), biology (e.g., protein structure prediction), chemistry, or image processing. We will focus on the first of these examples, natural language parsing. The literature on such problems clearly indicates that good accuracy cannot be achieved without models which capture the generalisations which are only reflected in the output structure. For example, in English sentences, if a noun is parsed as the subject of a verb, then these words must be compatible in their singular/plural markings, independent of whether they are near each other in the input sentence.

In addition to limiting the scope of this article to parsing problems, we focus on tasks where the training data specifies the output structure, but the labelling of this structure is not fully annotated. While the unannotated labelling may not be evaluated in the task, by assuming incomplete labelling we allow our models to capture generalisation which are not directly reflected in the labelled output structure given for training. For example, the training data for natural language parsing problems is generally assumed to be a tree, but assuming that all generalisations can be expressed in terms of one-level fragments of the tree leads to poor empirical performance. However, much better performance can be achieved with such a model by extending the labelling to include features of the structural context (Charniak, 2000). Because we want to learn the necessary additional labelling, we need to solve a limited form of grammar induction.

Graphical models provide the formal mechanisms needed to learn and reason about incomplete labelling, using latent variables. They also provide the formal mechanisms needed to specify the statistical dependencies implied by the structure of a single output category. However, these mechanisms are not sufficient to specify a complete probability model for a parsing problem, because we need to specify the statistical dependencies for the complete space of possible output categories. As we will discuss in Section 3, even graphical models for unbounded sequence labelling, such as dynamic Bayesian networks, are in general not adequate for this task, because they are limited to finite-state models.

There are well established methods for specifying probabilistic models of parsing problems, based on grammar formalisms, such as probabilistic context-free grammars (PCFGs). The grammar formalism defines how the complete space of possible pairs of an input sequence with an output structure can be specified as a set of sequences of decisions about the input-output pair. Each possible sequence of decisions, called a derivation, specifies a single input-output pair (e.g., phrase structure tree or protein structure). The probability model is then defined in terms of probabilities for each decision. In its most general form, these decision probabilities are conditioned on anything from the unbounded history of previous decisions:

$$P(T) = P(D^1, ..., D^m) = \prod_t P(D^t | D^1, \ldots, D^{t-1}), \tag{1}$$

where T is the input-output structure and $D^1, \ldots, D^m$ is its equivalent sequence of decisions.

In PCFGs, the context-free assumption means that only a bounded amount of the history $D^1, \ldots, D^{t-1}$ is relevant to the probability for decision D^t. The context-free assumption only allows statistical dependencies within each bounded one-level subtree of the output tree, so two such subtrees can only interact through the bounded choice of label for the node they share, if any. Because the context-free assumption is defined in terms of the output structure, not in terms of the

input sequence, which decisions in the history are relevant depends on the output structure specified by the derivation. In graphical models, such a specification of which decisions are statistically dependent on which other decisions is called the "model structure". Thus PCFGs, like other grammar formalisms, are examples of models where the model structure is a function of the output structure, not just of the input sequence. This is the fundamental distinction between models of parsing problems and models of sequence labelling problems, and it will be central to our discussions in this article.

The most common approach to building probability models for parsing problems is to use PCFGs without any latent variables (e.g., Charniak, 2000; Collins, 1999; Durbin et al., 1998), but this approach relies on hand-built sets of features to represent the unbounded decision histories in (1). Latent probabilistic context-free grammars (LPCFGs) (Matsuzaki et al., 2005) extend the node labels of PCFGs with latent annotations, but previous proposals have successfully induced only a small number of latent annotations.

An alternative proposal to extending the labelling of parse trees is to use the hidden units of a neural network (Henderson, 2003). In the model of Henderson (2003), vectors of hidden unit values decorate the positions t in the derivation sequence, and are used to encode features of the unbounded derivation history $D^1, \ldots, D^{t-1}$. As with LPCFGs, the pattern of interdependencies between layers of hidden units is a function of the output structure, making it appropriate for parsing problems. But unlike LPCFGs, the pattern of interdependencies is not required to respect the context-free assumption. This model achieved state-of-the-art results, but there is no clear probabilistic semantics for the induced hidden representations.

In this article, we propose a class of graphical models which we call incremental sigmoid belief networks (ISBNs), which are closely related to the neural network of Henderson (2003), but which have a clear probabilistic semantics for all their variables. ISBNs are a kind of sigmoid belief network (Neal, 1992), but are dynamic models and have an incrementally specified set of statistical dependencies. Each position in the decision sequence has a vector of latent state variables, which are statistically dependent on variables from previous positions via a pattern of edges determined by the previous decisions. This incrementally specified model structure allows ISBNs to capture the generalisations which are only reflected in the output structure, such as the tendency towards correlations which are local in the output structure, which motivates the context-free assumption of PCFGs.

Allowing the model structure to depend on the output structure means that the complete model structure is not known until the complete output derivation is known. In general, this can complicate decoding (i.e., parsing) because computing probabilities for sub-derivations requires marginalising out the unknown portion of the model structure, which in the worst case could require summing over an unbounded number of possible model structures. The properties of ISBNs avoid this problem because the probability of a derivation prefix is always independent of the unknown portion of the model structure, as discussed in Section 3.

Despite this simplification, exact inference (i.e., computing probabilities) is not in general tractable in ISBNs, because they allow large vectors of latent variables in a heavily interconnected directed model. We demonstrate the practical applicability of ISBN models by providing efficient approximate inference methods. We consider two forms of approximation for ISBNs, a feed-forward neural network approximation (NN) and a form of mean field approximation (Saul and Jordan, 1999). In Section 5, we first show that the neural network model in Henderson (2003) can be viewed as a coarse approximation to inference with ISBNs. We then propose an incremental

mean field method (IMF), which provides an improved approximation but remains tractable. Both these approximations give us valid probability models.

In Section 7, we present two empirical evaluations. In the first experiment, we trained both of the approximation models on artificial data generated from random ISBNs. The NN model achieves a 60% average relative error reduction over a baseline model and the IMF model achieves a further 27% average relative error reduction over the NN model. These results demonstrate that the distribution of output structures specified by an ISBN can be approximated, that these approximations can be learnt from data, and that the IMF approximation is indeed better than the NN approximation. In the second experiment, we apply both of the approximation models to phrase structure parsing with data from the Wall Street Journal Penn Treebank (Marcus et al., 1993). The IMF model achieves statistically significant error reduction of about 8% over the NN model. Results of the IMF model are non-significantly worse (less than 1% relative error increase) than the results of one of the best known history-based models of parsing (Charniak, 2000). We argue that this correlation between better approximation and better accuracy suggests that ISBNs are a good abstract model for structured prediction.

Section 8 discusses related work not covered in the rest of this article. It focuses particularly on previous work on LPCFGs.

2. Inference with Sigmoid Belief Networks

Before defining ISBNs, we provide background on sigmoid belief networks. A sigmoid belief network (SBN) (Neal, 1992) is a type of Bayesian network. Bayesian networks are directed acyclic graphs where the nodes are variables and the edges specify statistical dependencies between variables. SBNs have binary variables which have conditional probability distributions (CPDs) of the form:

$$P(S_i = 1|Par(S_i)) = \sigma(\sum_{S_j \in Par(S_i)} J_{ij} S_j), \tag{2}$$

where $Par(S_i)$ is the set of variables with edges directed to S_i, σ denotes the logistic sigmoid function $\sigma(x) = 1/(1+e^{-x})$, and J_{ij} is the weight for the edge from variable S_j to variable S_i.[1] SBNs are similar to feed-forward neural networks, but unlike neural networks, SBNs have a precise probabilistic semantics of their hidden variables. In ISBNs we consider a generalised version of SBNs where we allow variables with any range of discrete values. The normalised exponential function is used to define the CPDs at these nodes:

$$P(S_i = k|Par(S_i)) = \frac{\exp(\sum_{S_j \in Par(S_i)} W^i_{kj} S_j)}{\sum_{k'} \exp(\sum_{S_j \in Par(S_i)} W^i_{k'j} S_j)}, \tag{3}$$

where W^i is the weight matrix for the variable S_i.

Exact inference with all but very small SBNs is not tractable. Initially sampling methods were used (Neal, 1992), but they are also not feasible for large networks, especially for the dynamic models of the type described in Section 4. Variational methods have also been proposed for approximating SBNs (Saul et al., 1996; Saul and Jordan, 1999). The main idea of variational methods (Jordan

1. For convenience, where possible, we will not explicitly include bias terms in expressions, assuming that every latent variable in the model has an auxiliary parent variable set to 1.

et al., 1999) is, roughly, to construct a tractable approximate model with a number of free parameters. The free parameters are set so that the resulting approximate model is as close as possible to the original model for a given inference problem.

The simplest example of a variational method is the mean field method, originally introduced in statistical mechanics and later applied to neural networks in Hinton et al. (1995). Let us denote the set of visible variables in the model by V and latent (hidden) variables by $H = h_1, \ldots, h_l$. The mean field method uses a fully factorised distribution $Q(H|V) = \prod_i Q_i(h_i|V)$ as the approximate model, where each Q_i is the distribution of an individual latent variable. The independence between the variables h_i in this approximate distribution Q does not imply independence of the free parameters which define the Q_i. These parameters are set to minimise the Kullback-Leibler divergence between the approximate distribution $Q(H|V)$ and the true distribution $P(H|V)$ or, equivalently, to maximise:

$$L_V = \sum_H Q(H|V) \ln \frac{P(H,V)}{Q(H|V)}. \tag{4}$$

The expression L_V is a lower bound on the log-likelihood $\ln P(V)$. It is used in the mean field theory (Saul and Jordan, 1999) as an approximation of the log-likelihood. However, in our case of dynamic graphical models, as explained later, we have to use a different approach which allows us to construct an incremental structured prediction method without needing to introduce the additional parameters proposed in Saul and Jordan (1999), as we will discuss in Section 5.3.

3. Incrementally Specifying Model Structure

We want to extend SBNs to make them appropriate for modelling parsing problems. As discussed in the introduction, this requires being able to model arbitrarily long decision sequences $D^1, ..., D^m$, and being able to specify the pattern of edges (the model structure) as a function of the chosen output structure. In this section, we define how incremental sigmoid belief networks specify such model structures.

To extend SBNs for processing arbitrarily long sequences, such as the derivation decision sequence $D^1, ..., D^m$, we use dynamic models. This gives us a form of dynamic Bayesian network (DBN). To handle unboundedly long sequences, DBNs specify a Bayesian network template which gets instantiated for each position in the sequence, thereby constructing a Bayesian network which is as large as the sequence is long. This constructed Bayesian network is illustrated in the rightmost graph of Figure 1, where the repeated two-box pattern is the template, and the left-to-right order is the derivation order. This template instantiation defines a new set of variables for each position in the sequence, but the set of edges and parameters for these variables are the same as in other positions. The edges which connect variables instantiated for different positions must be directed forward in the sequence, thereby allowing a temporal interpretation of the sequence. DBNs based on sigmoid belief networks were considered in Sallans (2002) in the context of reinforcement learning. Normally, DBNs only allow edges between adjacent (or a bounded window of) positions, which imposes a Markov assumption on statistical dependencies in the Bayesian network.

The problem with only allowing edges between variables instantiated at positions which are adjacent (or local) in the decision sequence is that this does not allow the model structure to adequately reflect the correlations found in parsing problems. In particular, in many domains, correlations tend to be local in the output structure, even when they are not local in the derivation sequence for that structure. To capture these correlations in the statistical dependencies learnt by the model, we want

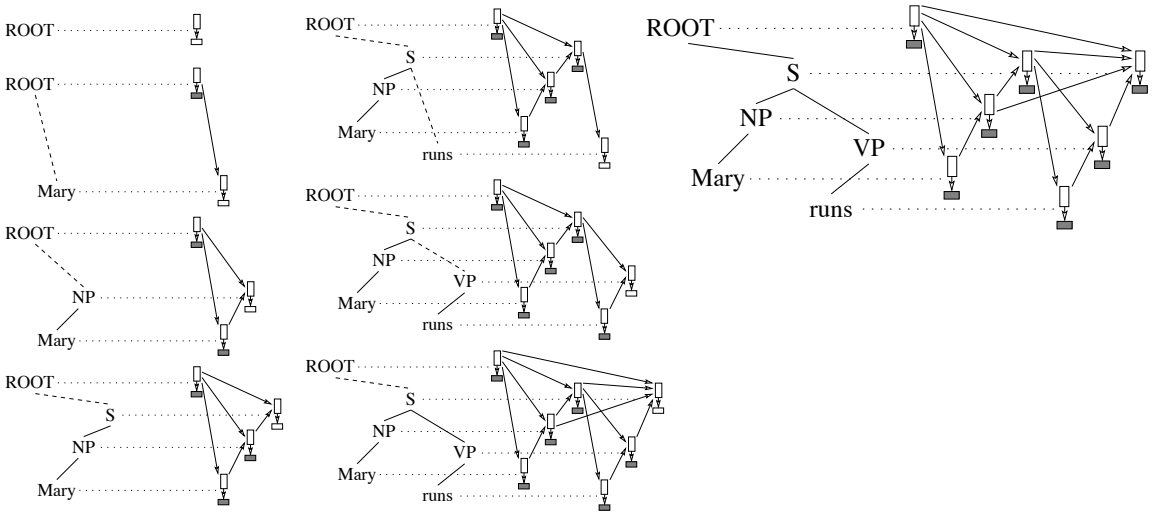


Figure 1: Illustration of the predictive LR derivation of an output structure and its associated incremental specification of an ISBN model structure (ordered top-to-bottom, left-to-right). Dotted lines indicate the top of the parser's stack at each derivation decision in the model structure.

the edges of the model to reflect locality in the output structure. This requires specifying edges based on the actual outputs in the decision sequence $D^1, ..., D^m$, not just based on adjacency in this sequence.

We constrain this edge specification so that a decision $D^{t'}$ can only effect the placement of edges whose destination variable is at a position $t > t'$ after the decision $D^{t'}$. This gives us a form of switching model (Murphy, 2002), where each decision switches the model structure used for the remaining decisions. We allow the incoming edges for a given position to be any discrete function of the sequence of decisions which precede that position. For this reason, we call our model an "incremental" model, not just a dynamic model; the structure of the Bayesian network is determined incrementally as the decision sequence proceeds. This incremental specification of the model structure is illustrated in Figure 1 (the directed graphs), along with the partial output structures incrementally specified by the derivation (the trees). In Figure 1, dotted lines associate a position's instantiated template with the node in the output structure which is on top of the parser's stack when making that position's decision. Note that the incoming edges for a position's instantiated template reflect edges between the associated nodes in the partial output structure.

Any discrete function can be used to map the preceding sequence of decisions to a set of incoming edges for a given decision. In general, we can characterise this function in terms of an automaton which reads derivations and deterministically outputs model structures. For every derivation prefix $D^1, ..., D^{t-1}$, the automaton outputs a set of labelled positions in the derivation prefix. For each labelled position $(t-c, r)$ in this set, label r determines which variables instantiated at that position are linked to which variables instantiated at the current position t, and with which parameters.[2] For

2. In all our models to date, we have respected the additional constraint that there is at most one labelled position in the set for each label r, so the size of the set is bounded. We do not impose this constraint here because the model is still well defined without it, but we do not have empirical evidence about the effect of removing it.

example, the ISBN illustrated in Figure 1 uses a push-down automaton to compute which output structure nodes are currently important (e.g., the top and next-to-top nodes on the automaton's stack) and specifies conditional dependencies between the current decision and previous decisions where these nodes were on the top of the stack. By using a push-down automaton, this model is able to express non-Markovian (e.g., context-free) regularities in the derivation sequences.

Previous applications of switching models to DBNs (e.g., Murphy, 2002) have allowed statistical dependencies to be a function of the output, but only of the output from the immediately preceding position in the sequence, and therefore have only allowed switching between a bounded number of alternatives. Because the number of switched alternatives is bounded, the whole set of alternatives can be expressed as a single bounded model, whose CPDs incorporate the discrete switching. Thus, switching does not allow us to specify any models which could not be specified with a complicated DBN, so switching DBNs also impose some form of Markov assumption. In terms of the automata discussed above, this means that switching DBNs can be expressed using finite-state automata, so would only be appropriate for problems with a regular-language structure to their output categories. This limitation does not give us sufficient power to express the kinds of output-conditioned statistical dependencies we need for parsing problems in general. Therefore, it is crucial to distinguish between standard dynamic models and our incremental models.

Incremental sigmoid belief networks allow the model structure to depend on the output structure without overly complicating the inference of the desired conditional probabilities $P(D^t|D^1,\ldots,D^{t-1})$. Computing this probability requires marginalising out the unknown model structure for the portion of the Bayesian network which follows position t. In general, this could require explicitly summing over multiple possible model structures, or in the worst case summing over the unbounded number of possible model structures. ISBNs avoid summing over any of these possible model structures because in ISBNs $P(D^t|D^1,\ldots,D^{t-1})$ is independent of all model structure which follows position t. This can be proved by considering two properties of ISBNs. At position t in the sequence, the only edges whose placement are not uniquely determined by $D^1,\ldots,D^{t-1}$ have their destinations after t. Also, none of the variables after t are visible (i.e., have their values specified in $D^1,\ldots,D^{t-1}$). Therefore none of the edges whose placement is not yet known can have any impact on the inference of $P(D^t|D^1,\ldots,D^{t-1})$, as follows directly from well known properties of Bayesian networks. This property implies that each individual Bayesian network depicted in Figure 1 can be used to compute the conditional probability of its next derivation decision, and it will give the same answer as if the same conditional probability were computed in the final Bayesian network at the end of the derivation, or indeed in any such valid continuation.

The use of directed edges to avoid the need to sum over unknown model structures can also be seen in Hidden Markov Models (HMMs). Given a sequence prefix, we can use an HMM to infer the probability of the following element of the sequence. This distribution is not dependent on the total length of the sequence, which would be needed to draw the complete HMM model for the sequence. Note that this property does not hold for undirected graphical models, such as Conditional Random Fields (Lafferty et al., 2001). Rohanimanesh et al. (2009) investigate inference in undirected models with edges that are a function of the output structure, but the solutions are approximate and computationally expensive.

The incremental specification of model structure can also be seen in LPCFGs. Given a top-down left-to-right derivation of a phrase structure tree, the dependencies between LPCFG derivation decisions have the same structure as the phrase structure tree, but with LPCFG rules (one-level subtrees) labelling each node of this derivation tree. The number of branches at a node in the

derivation tree is determined by the rule which is chosen to label that node, thereby incrementally specifying the complete derivation tree. If we expressed an LPCFG as a graphical model, the model structures would have the same general form as the derivation trees, so the model structure would also be incrementally specified. Also, the edges in this graphical model would need to be directed, because LPCFG rule probabilities are locally normalised. Therefore LPCFG can also be thought of as Bayesian networks with incrementally specified model structure. The differences between ISBNs and LPCFG will be discussed in the next section and Section 8.

As illustrated by the above examples, the argument for the incremental specification of model structure can be applied to any Bayesian network architecture, not just sigmoid belief networks. We focus on ISBNs because, as shown in Section 5, they are closely related to the empirically successful neural network models of Henderson (2003). This previous work has shown that the combination of logistic sigmoid hidden units and having a model structure which reflect locality in the output structure results in a powerful form of feature induction. The edges from hidden units to hidden units allow information to propagate beyond the notion of immediate structural locality defined in the model, but the logistic sigmoid ensures a bias against propagating information through long chains of hidden units, thereby providing a soft but domain-appropriate bias to feature induction.

4. The Probabilistic Model of Structured Prediction

In this section we complete the definition of incremental sigmoid belief networks for grammar learning. We only consider joint probability models, since they are generally simpler and, unlike history-based conditional models, do not suffer from the label bias problem (Bottou, 1991). Also, in many complex predication tasks, such as phrase structure parsing, many of the most accurate models make use of a joint model, either in reranking or model combinations (e.g., Charniak and Johnson, 2005; Henderson, 2004).

We use a history-based probability model, as in Equation (1), but instead of treating each D^t as an atomic decision, it will be convenient below to further split it into a sequence of elementary decisions $D^t = d^t_1, \ldots, d^t_n$:

$$P(D^t | D^1, \ldots, D^{t-1}) = \prod_k P(d^t_k | h(t,k)),$$

where $h(t,k)$ denotes the decision history $D^1, \ldots, D^{t-1}, d^t_1, \ldots, d^t_{k-1}$. For example, a decision to create a new node in a labelled output structure can be divided into two elementary decisions: deciding to create a node and deciding which label to assign to it.

An example of the kind of graphical model we propose is illustrated in Figure 2. It is organised into vectors of variables: latent state variable vectors $S^{t'} = s^{t'}_1, \ldots, s^{t'}_n$, representing an intermediate state at position t', and decision variable vectors $D^{t'}$, representing a decision at position t', where $t' \leq t$. Variables whose value are given at the current decision (t,k) are shaded in Figure 2; latent and current decision variables are left unshaded.

As illustrated by the edges in Figure 2, the probability of each state variable s^t_i depends on all the variables in a finite set of relevant previous state and decision vectors, but there are no direct dependencies between the different variables in a single state vector. As discussed in Section 3, this set of previous state and decision vectors is determined by an automaton which runs over the derivation history $D^1, \ldots, D^{t-1}$ and outputs a set of labelled positions in the history which are connected to the current position t. For each pair $(t-c, r)$ in this set, r represents a relation between

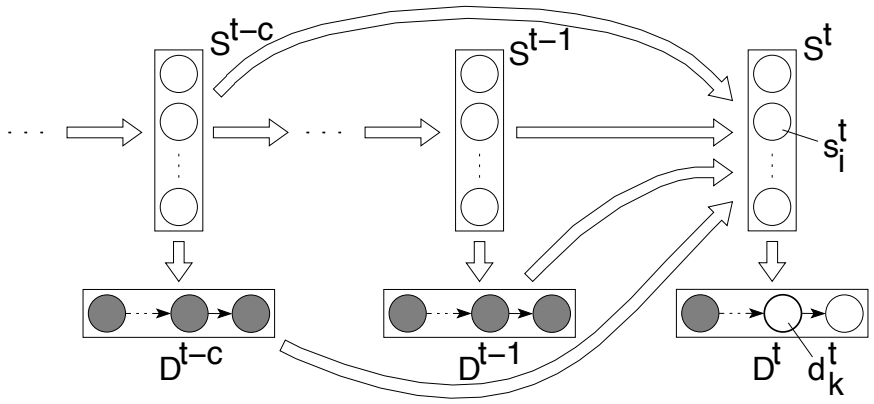


Figure 2: ISBN for estimating $P(d_k^t|h(t,k))$.

position t and the position $t-c$ in the history. We denote by $r(t-c,t)$ the predicate which returns true if the position $t-c$ with the relation label r is included in the set for t, and false otherwise. In general this automaton is allowed to perform arbitrary computations, as specified by the model designer. For example, it could select the most recent state where the same output structure node was on the top of the automaton's stack, and a decision variable representing that node's label. Each such selected relation r has its own distinct weight matrix for the resulting edges in the graph, but the same weight matrix is used at each position where the relation is relevant (see Section 7.2 for examples of relation types we use in our experiments).

We can write the dependency of a latent variable s_i^t on previous latent variable vectors and a decision history as:

$$P(s_i^t = 1|S^1,\ldots,S^{t-1},h(t,1)) = \sigma\left(\sum_{r,t':r(t',t)}\sum_j J_{ij}^r s_j^{t'} + \sum_k B_{id_k^{t'}}^{rk}\right), \tag{5}$$

where J^r is the latent-to-latent weight matrix for relation r and B^{rk} is the decision-to-latent weight matrix for relation r and elementary decision k. If there is no previous step $t' < t$ which is in relation r to the time step t, that is, $r(t',t)$ is false for all t', then the corresponding relation r is skipped in the summation. For each relation r, the weight J_{ij}^r determines the influence of the jth variable in the related previous latent vector $S^{t'}$ on the distribution of the ith variable of the considered latent vector S^t. Similarly, $B_{id_k^{t'}}^{rk}$ defines the influence of the past decision $d_k^{t'}$ on the distribution of the considered latent vector variable s_i^t.

In the previous paragraph we defined the conditional distribution of the latent vector variables. Now we describe the distribution of the decision vector $D^t = d_1^t,\ldots d_n^t$. As indicated in Figure 2, the probability of each elementary decision d_k^t depends both on the current latent vector S^t and on the previously chosen elementary action d_{k-1}^t from D^t. This probability distribution has the normalised exponential form:

$$P(d_k^t = d|S^t,d_{k-1}^t) = \frac{\Phi_{h(t,k)}(d)\exp(\sum_j W_{dj}s_j^t)}{\sum_{d'}\Phi_{h(t,k)}(d')\exp(\sum_j W_{d'j}s_j^t)}, \tag{6}$$

where $\Phi_{h(t,k)}$ is the indicator function of the set of elementary decisions that can possibly follow the last decision in the history $h(t,k)$, and the W_{dj} are the weights of the edges from the state variables. Φ is essentially switching the output space of the elementary inference problems $P(d_k^t = d|S^t, d_{k-1}^t)$ on the basis of the previous decision d_{k-1}^t. For example, in a generative history-based model of natural language parsing, if decision d_1^t was to create a new node in the tree, then the next possible set of decisions defined by $\Phi_{h(t,2)}$ will correspond to choosing a node label, whereas if decision d_1^t was to generate a new word then $\Phi_{h(t,2)}$ will select decisions corresponding to choosing this word.

Given this design for using ISBNs to model derivations, we can compare such ISBN models to LPCFG models. As we showed in the previous section, LPCFGs can also be thought of as Bayesian networks with incrementally specified model structure. One difference between LPCFGs and ISBNs is that LPCFGs add latent annotations to the symbols of a grammar, while ISBNs add latent annotations to the states of an automaton. However, this distinction is blurred by the use of grammar transforms in LPCFG models, and the many equivalences between grammars and automata. But certainly, the automata of ISBNs are much less constrained than the context-free grammars of LPCFGs. Another distinction between LPCFGs and ISBNs is that LPCFGs use latent annotations to split symbols into multiple atomic symbols, while ISBNs add vectors of latent variables to the existing symbol variables. The structure of the similarities between vectors is much richer than the structure of similarities between split atomic symbols, which gives ISBNs a more structured latent variable space than LPCFGs. This makes learning easier for ISBNs, allowing the induction of more informative latent annotations. Both these distinctions will be discussed further in Section 8.

5. Approximating Inference in ISBNs

Exact inference with ISBNs is straightforward, but not tractable. It involves a summation over all possible variable values for all the latent variable vectors. The presence of fully connected latent variable vectors does not allow us to use efficient belief propagation methods. Even in the case of dynamic SBNs (i.e., Markovian models), the large size of each individual latent vector would not allow us to perform the marginalisation exactly. This makes it clear that we need to develop methods for approximating the inference problems required for structured prediction. Standard Gibbs sampling (Geman and Geman, 1984) is also expensive because of the huge space of variables and the need to resample after making each new decision in the sequence. It might be possible to develop efficient approximations to Gibbs sampling or apply more complex versions of Markov Chain Monte-Carlo techniques, but sampling methods are generally not as fast as variational methods. In order to develop sufficiently fast approximations, we have investigated variational methods.

This section is structured as follows. We start by describing the application of the standard mean field approximation to ISBNs and discuss its limitations. Then we propose an approach to overcome these limitations, and two approximation methods. First we show that the neural network computation used in Henderson (2003) can be viewed as a mean field approximation with the added constraint that computations be strictly incremental. Then we relax this constraint to build more accurate but still tractable mean field approximation.

5.1 Applicability of Mean Field Approximations

In this section we derive the most straightforward way to apply mean field methods to ISBN. Then we explain why this approach is not feasible for structured prediction problems of the scale of natural language parsing.

The standard use of the mean field theory for SBNs (Saul et al., 1996; Saul and Jordan, 1999) is to approximate probabilities using the value of the lower bound L_V from expression (4) in Section 2. To obtain a tighter bound, as we explained above, L_V is maximised by choosing the optimal distribution Q. To approximate $P(d_k^t|h(t,k))$ using the value of L_V, we have to include the current decision d_k^t in the set of visible variables, along with the visible variables specified in $h(t,k)$. Then to estimate the conditional probability $P(d_k^t|h(t,k))$, we need to normalise over the set of all possible value of d_k^t. Thus we need to compute a separate estimate $\max_Q L_V^{t,k}(d)$ for each possible value of $d_k^t = d$:

$$\max_Q L_V^{t,k}(d) = \max_Q \sum_H Q(H^t|h(t,k),d_k^t = d) \ln \frac{P(H^t,h(t,k),d_k^t = d)}{Q(H^t|h(t,k),d_k^t = d)},$$

where $H^t = \{S^1,\ldots,S^t\}$. Then $P(d_k^t = d|h(t,k))$ can be approximated as the normalised exponential of $L_V^{t,k}(d)$ values:

$$\hat{P}(d_k^t = d|h(t,k)) = \frac{\exp(\max_Q L_V^{t,k}(d))}{\sum_{d'} \exp(\max_Q L_V^{t,k}(d'))}. \tag{7}$$

It is not feasible to find the optimal distribution Q for SBNs, and mean field methods (Saul et al., 1996; Saul and Jordan, 1999) use an additional approximation to estimate $\max_Q L_V^{t,k}(d)$. Even with this approximation, the maximum can be found only by using an iterative search procedure. This means that decoding estimator (7) requires performing this numerical procedure for every possible value of the next decision. Unfortunately, in general this is not feasible, in particular with labelled output structures where the number of possible alternative decisions d_k^t can be large. For our generative model of natural language parsing, decisions include word predictions, and there can be a very large number of possible next words. Even if we choose not to recompute mean field parameters for all the preceding states $S^{t'}, t' < t$, but only for the current state S^t (as proposed below), tractability still remains a problem.[3]

In our modifications of the mean field method, we propose to consider the next decision d_k^t as a hidden variable. Then the assumption of full factorisability of $Q(H^t,d_k^t|h(t,k))$ is stronger than in the standard mean field theory because the approximate distribution Q is no longer conditioned on the next decision d_k^t. The approximate fully factorisable distribution $Q(H|V)$ can be written as:

$$Q(H|V) = q_k^t(d_k^t) \prod_{t'i} \left(\mu_i^{t'}\right)^{s_i^{t'}} \left(1-\mu_i^{t'}\right)^{1-s_i^{t'}}.$$

where $\mu_i^{t'}$ is the free parameter which determines the distribution of state variable i at position t', namely its mean, and $q_k^t(d_k^t)$ is the free parameter which determines the distribution over decisions d_k^t. Importantly, we use $q_k^t(d)$ to estimate the conditional probability of the next decision:

$$P(d_k^t = d|h(t,k)) \approx q_k^t(d),$$

3. We conducted preliminary experiments with natural language parsing on very small data sets and even in this setup the method appeared to be very slow and, surprisingly, not as accurate as the modification considered further in this section.

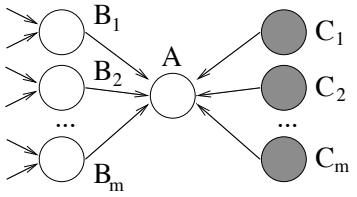


Figure 3: A graphical model fragment where variable A is a sink.

and the total structure probability is therefore computed as the product of decision probabilities corresponding to its derivation:

$$P(T) = P(D^1, \ldots, D^m) \approx \prod_{t,k} q_k^t(d_k^t). \tag{8}$$

5.2 A Feed-Forward Approximation

In this section we will describe the sense in which neural network computation can be regarded as a mean field approximation under an additional constraint of strictly feed-forward computation. We will call this approximation the feed-forward approximation. As in the mean field approximation, each of the latent variables in the feed-forward approximation is independently distributed. But unlike the general case of mean field approximation, in the feed-forward approximation we only allow the parameters of every distribution $Q(s_i^{t'}|h(t,k))$ and $Q(d_k^t|h(t,k))$ to depend on the approximate distributions of their parents, thus requiring that any information about the distribution of its descendants is not taken into account. This additional constraint increases the potential for a large KL divergence with the true model, but it significantly simplifies the computations.

We start with a simple proposition for general graphical models. Under the feed-forward assumption, computation of the mean field distribution of a node in an ISBN is equivalent to computation of a distribution of a variable corresponding to a sink in the graph of the model, that is, a node which does not have any outgoing edges. For example, node A is a sink in Figure 3. The following proposition characterises the mean field distribution of a sink.

Proposition 1 *The optimal mean field distribution of a sink A depends on the mean field distribution $Q(B)$ of its hidden parents $B = (B_1, \ldots, B_m)$ as*

$$Q(A = a) \propto \exp(E_Q \log P(A = a|B,C)),$$

where Q is the mean field distribution of hidden variables, P is the model distribution, C are visible parents of the node A and E_Q denotes the expectation under the mean field distribution $Q(B)$.

This proposition is straightforward to prove by maximising the variational bound L_V (4) with respect to the distribution $Q(A)$. Now we can use the fact that SBNs have log-linear CPD. By substituting their CPD given in expression (2) for P in the lemma statement, we obtain:

$$Q(S_i = 1) = \sigma(\sum_{S_j \in Par(S_i)} J_{ij}\mu_j),$$

which exactly replicates computation of a feed-forward neural network with the logistic sigmoid activation function. Similarly, we can show that for variables with soft-max CPD, as defined in (3), their mean field distribution will be the log-linear function of their parents' means. Therefore minimising KL divergence under the constraint of feed-forward computation is equivalent to using log-linear functions to compute distributions of random variables given means of their parents.

Now let us return to the derivation of the feed-forward approximation of ISBNs. As we just derived, under the feed-forward assumption, means of the latent vector $S^{t'}$ are given by

$$\mu_i^{t'} = \sigma(\eta_i^{t'}),$$

where $\eta_i^{t'}$ is the weighted sum of the parent variables' means:

$$\eta_i^{t'} = \sum_{r,t'':r(t'',t')} \sum_j J_{ij}^r \mu_j^{t''} + \sum_k B_{id_k^{t''}}^{rk}, \tag{9}$$

as follows from the definition of the corresponding CPD (5).

The same argument applies to decision variables; the approximate distribution of the next decision $q_k^t(d)$ is given by

$$q_k^t(d) = \frac{\Phi_{h(t,k)}(d) \exp(\sum_j W_{dj} \mu_j^t)}{\sum_{d'} \Phi_{h(t,k)}(d') \exp(\sum_j W_{d'j} \mu_j^t)}. \tag{10}$$

The resulting estimate of the probability of the entire structure is given by (8).

This approximation method replicates exactly the computation of the feed-forward neural network model of Henderson (2003), where the above means $\mu_i^{t'}$ are equivalent to the neural network hidden unit activations. Thus, that neural network probability model can be regarded as a simple approximation to the ISBN graphical model.

In addition to the drawbacks shared by any mean field approximation method, this feed-forward approximation cannot capture bottom-up reasoning. By bottom-up reasoning, we mean the effect of descendants in a graphical model on distributions of their ancestors. For mean field approximations to ISBNs, it implies the need to update the latent vector means $\mu_i^{t'}$ after observing a decision d_k^t, for $t' \leq t$. The use of edges directly from decision variables to subsequent latent vectors is designed to mitigate this limitation, but such edges cannot in general accurately approximate bottom-up reasoning. The next section discusses how bottom-up reasoning can be incorporated in the approximate model.

5.3 Incremental Mean Field Approximation

In this section we relax the feed-forward assumption to incorporate bottom-up reasoning into the approximate model. Again as in the feed-forward approximation, we are interested in finding the distribution Q which maximises the quantity L_V in expression (4). The decision distribution $q_k^t(d_k^t)$ maximises L_V when it has the same dependence on the latent vector means μ_j^t as in the feed-forward approximation, namely expression (10). However, as we mentioned above, the feed-forward computation does not allow us to compute the optimal values of state means $\mu_i^{t'}$.

Optimally, after each new decision d_k^t, we should recompute all the means $\mu_i^{t'}$ for all the latent vectors $S^{t'}$, $t' \leq t$. However, this would make the method intractable for tasks with long decision sequences. Instead, after making each decision d_k^t and adding it to the set of visible variables

V, we recompute only the means of the current latent vector S^t. This approach also speeds up computation because, unlike in the standard mean field theory, there is no need to introduce an additional variational parameter for each hidden layer variable s_i^t.

The denominator of the normalised exponential function in (6) does not allow us to compute L_V exactly. Instead, we approximate the expectation of its logarithm by substituting S_j^t with their means:[4]

$$E_Q \ln \sum_d \Phi_{h(t,k)}(d) \exp(\sum_j W_{dj} S_j^t) \approx \ln \sum_d \Phi_{h(t,k)}(d) \exp(\sum_j W_{dj} \mu_j^t),$$

where the expectation is taken over the latent vector S^t distributed according to the approximate distribution Q. Unfortunately, even with this assumption there is no analytic way to maximise the approximation of L_V with respect to the means μ_i^t, so we need to use numerical methods. We can rewrite the expression (4) as follows, substituting the true $P(H,V)$ defined by the graphical model and the approximate distribution $Q(H|V)$, omitting parts independent of the means μ_i^t:

$$\begin{aligned} L_V^{t,k} = & \sum_i -\mu_i^t \ln \mu_i^t - (1-\mu_i^t) \ln \left(1-\mu_i^t\right) + \mu_i^t \eta_i^t \\ & + \sum_{k'<k} \sum_j W_{d_{k'}^t j} \mu_j^t - \ln \left(\sum_d \Phi_{h(t,k')}(d) \exp(\sum_j W_{dj} \mu_j^t) \right), \end{aligned} \tag{11}$$

here, η_i^t is computed from the previous relevant state means and decisions as in (9). This expression is concave with respect to the parameters μ_i^t, so the global maximum can be found. In the appendix, where we derive the learning algorithm, we show that the Hessian of this expression can be viewed as the negated sum of a positive diagonal matrix and some covariance matrices, thus implying the concavity of expression (11). We use coordinatewise ascent, where each μ_i^t is selected by a line search (Press et al., 1996) while keeping other $\mu_{i'}^t$ fixed.

Though we avoided re-computation of means of the previous states, estimation of the complex decision probability $P(D^t|h(t,k))$ will be expensive if the decision D^t is decomposed in a large number of elementary decisions. As an example, consider a situation in dependency parsing, where after deciding to create a link, the parser might need to decide on the type of the link and, then, predict the part of speech type of the word and, finally, predict the word itself. The main reason for this complexity is the presence of the summation over k' in expression (11), which results in expensive computations during the search for an optimal value of μ_i^t. This computation can be simplified by using the means of S^t computed during the estimation of $P(d_{k-1}^t|h(t,k-1))$ as priors for the computation of the same means during the estimation of $P(d_k^t|h(t,k))$. If we denote the means computed at an elementary step (t,k) as $\mu_i^{t,k}$, then for $k=1$, minimisation of $L_V^{t,k}$ can be performed analytically, by setting

$$\mu_i^{t,1} = \sigma(\eta_i^t). \tag{12}$$

4. In initial research, we considered the introduction of additional variational parameters associated with every possible value of the decision variable in a way similar to Saul and Jordan (1999), but this did not improve the prediction accuracy of the model, and considerably increased the computational time.

For $k > 1$, expression (11) can be rewritten as:

$$\begin{aligned} L_V^{t,k} = & \sum_i -\mu_i^{t,k} \ln \mu_i^{t,k} - (1-\mu_i^{t,k}) \ln \left(1-\mu_i^{t,k}\right) \\ & + \mu_i^{t,k} \left(\ln \mu_i^{t,k-1} - \ln(1-\mu_i^{t,k-1})\right) + W_{d^t_{k-1} i} \mu_i^{t,k} \\ & - \ln \left(\sum_d \Phi_{h(t,k-1)}(d) \exp\left(\sum_j W_{dj} \mu_j^{t,k}\right) \right). \end{aligned} \tag{13}$$

Note that maximisation of this expression is done also after computing the last decision K_t for the state t. The resulting means μ^{t,K_t+1} are then used in the computation of $\eta_i^{t'}$ for the relevant future states t', that is, such t' that $r(t,t')$ holds for some r:

$$\eta_i^{t'} = \sum_{r,t:r(t,t')} \sum_j J^r_{ij} \mu_j^{t,K_t+1} + \sum_k B^{rk}_{id^t_k}, \tag{14}$$

Concavity of expression (13) follows from concavity of (11), as their functional forms are different by only a linear term and the presence of summation over the elementary decisions. See the appendix where we will show that the Hessian of $L_v^{t,k}$ is negative semidefinite, confirming this statement.

6. Learning and Decoding

We train the models described in Sections 5.2 and 5.3 to maximise the fit of the approximate models to the data. We use gradient descent, and a maximum likelihood objective function. In order to compute the derivatives with respect to the model parameters, the error should be propagated back through the structure of the graphical model. For the feed-forward approximation, computation of the derivatives is straightforward, as in neural networks (Rumelhart et al., 1986). But for the mean field approximation, this requires computation of the derivatives of the means μ_i^t with respect to the other parameters in expression (13). The use of a numerical search in the mean field approximation makes the analytical computation of these derivatives impossible, so a different method needs to be used to compute their values. The appendix considers the challenges arising when using maximum likelihood estimation with the incremental mean field algorithm and introduces a modification of the error backpropagation algorithm for this model. For both approximations, their respective backpropagation algorithms have computational complexity linear in the length of a derivation.

The standard mean field approach considered in Saul and Jordan (1999) maximised L_V (4) during learning, because L_V was used as an approximation of the log-likelihood of the training data. L_V is actually the sum of the log-likelihood and the negated KL divergence between the approximate distribution $Q(H|V)$ and the SBN distribution $P(H|V)$. Thus, maximising L_V will at the same time direct the SBN distribution toward configurations which have a lower approximation error. It is important to distinguish this regularisation of the approximate distribution from the Gaussian priors on the SBN parameters, which can be achieved by simple weight decay. We believe that these two regularisations should be complementary. However, in our version of the mean field method the approximate distributions of hidden decision variables q_k^t are used to compute the data likelihood (8) and, thus, maximising this target function will not automatically imply KL divergence minimisation. Application of an additional regularisation term corresponding to minimisation of the KL divergence might be beneficial for our approach, and it could be a subject of further research.

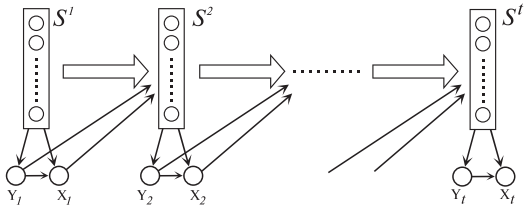


Figure 4: Dynamic SBN used in artificial experiments.

In our current experiments, we used standard weight decay, which regularises the SBN distribution with a Gaussian prior over weights.

ISBNs define a probability model which does not make any a-priori assumptions of independence between any decision variables. As we discussed in Section 3, the use of relations based on the partial output structure makes it possible to take into account statistical interdependencies between decisions closely related in the output structure, but separated by arbitrarily many positions in the input structure. In general, this property leads to the complexity of complete search being exponential in the number of derivation decisions. Fortunately, for many problems, such as natural language parsing, efficient heuristic search methods are possible.

7. Experiments

The goal of the evaluation is to demonstrate that ISBNs are an appropriate model for grammar learning. Also, we would like to show that learning the mean field approximation derived in Section 5.3 (IMF method) results in a sufficiently accurate model, and that this model is more accurate than the feed-forward neural network approximation (NN method) of Henderson (2003) considered in Section 5.2. First, we start with an artificial experiment where the training and testing data is known to have been generated by a SBN, and compare models based on each of the approximation methods. Second, we apply the models to a real problem, parsing of natural language, where we compare our approximations with state-of-the-art models.

7.1 Artificial Experiment

In order to have an upper bound for our artificial experiments, we do not consider incremental models, but instead use a dynamic sigmoid belief network, a first order Markovian model, and consider a sequence labelling task. This simplification allowed us to use Gibbs sampling from a *true* model as an upper bound of accuracy. The following generative story corresponds to the random dynamic SBNs:

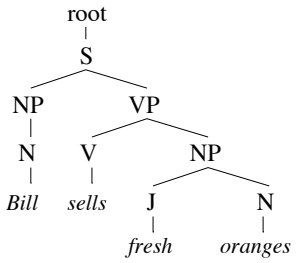


Figure 5: An example phrase structure tree.

Draw initial state vector S^1 from a distribution of initial states $P(S^1)$.
t = 0.
Do
t = t + 1,
draw a label Y^t from the distribution $P(Y^t|S^t)$ as in (6),
draw an input element X^t from the distribution $P(X^t|Y^t,S^t)$,
draw the next latent state vector S^{t+1} from $P(S^{t+1}|S^t,X^t,Y^t)$,
while $Y^t \neq 0$ and $t < t_{max}$.

A graphical representation of this dynamic model is shown in Figure 4. Different weight matrices were used in the computation of $P(X^t|Y^t,S^t)$ for each value of the label Y^t. It is easy to see that this model is a special case of the ISBN graphical model, namely Figure 2 with non-Markovian dependencies removed. The state vector length was set to 5, the number of possible labels to 6, the number of distinct input elements to 8, the maximal length of each sequence t_{max} to 100. We performed 10 experiments.[5] For each of the experiments, we trained both IMF and NN approximations on a training set of 20,000 elements, and tested them on another 10,000 elements. Weight-decay and learning rate were reduced through the course of the experiments whenever accuracy on the development set went down. Beam search with a beam of 10 was used during testing. The IMF methods achieved average error reduction of 27% with respect to the NN method, where accuracy of the Gibbs sampler was used as an upper bound (average accuracies of 80.5%, 81.0%, and 82.3% for the NN, IMF, and sampler, respectively).

The IMF approximation performed better than the NN approximation on 9 experiments out of 10 (statistically significant in 8 cases).[6] These results suggest that the IMF method leads to a much more accurate model than the NN method when the true distribution is defined by a dynamic SBN. In addition, the average relative error reduction of even the NN approximation over the unigram model exceeded 60% (the unigram model accuracy was 77.4% on average), which suggests that both approximations are sufficiently accurate and learnable.

7.2 Natural Language Parsing

We compare our two approximations on the natural language phrase structure parsing task. The output structure is defined as a labelled tree, which specifies the hierarchical decomposition of a

5. We preselected these 10 models to avoid random dynamic SBNs with trivial distributions. We excluded SBNs for which unigram model accuracy was within 3% of the Gibbs sampler accuracy, and where accuracy of the Gibbs sampler did not exceed 70%. All these constants were selected before conducting the experiments.
6. In all our experiments we used the permutation test (Diaconis and Efron, 1983) to measure significance and considered a result significant if p-value is below 5%.

Decisions	Stack
1. Shift $_{\text{Bill/N}}$	[root]
2. Project $_{\text{NP}}$	[root,Bill/N]
3. Project $_{\text{S}}$	[root,NP]
4. Shift $_{\text{sells/V}}$	[root,S]
5. Project $_{\text{VP}}$	[root, S, sells/V]
6. Shift $_{\text{fresh/J}}$	[root,S,VP]
7. Project $_{\text{NP}}$	[root,S,VP,fresh/J]
8. Shift $_{\text{oranges/N}}$	[root,S,VP,NP]
9.–12. Attach	[root,S,VP,NP,oranges/N],...,[root,S]

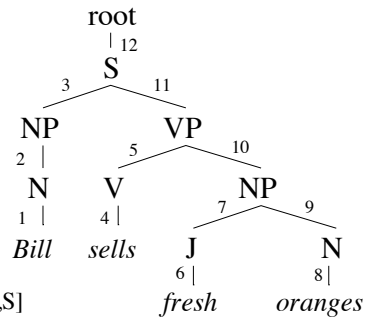


Figure 6: Derivation for a constituent parse tree.

sentence into phrases. An example of such a tree is presented on Figure 5, where the tree specifies that the adjective (J) *fresh* and the noun (N) *oranges* form a noun phrase (NP) *"fresh oranges"*, which, when combined with the verb (V) *sells*, forms the verb phrase (VP) *"sells fresh oranges"*. The hypothesis we wish to test here is that the more accurate approximation of ISBNs will result in a more accurate model of parsing. If this is true, then it suggests that ISBNs are a good abstract model for problems similar to natural language parsing, namely parsing problems which benefit from latent variable induction.

We replicated the same definition of derivation and the same pattern of interconnection between states as described in Henderson (2003). For the sake of completeness we will provide a brief description of the structure of the model here, though more details can be found in Henderson (2003).

The model uses a modification of the predictive LR order (Soisalon-Soininen and Ukkonen, 1979), illustrated in Figure 6. In this ordering, a parser decides to introduce a node into the parse tree after the entire subtree rooted at the node's first child has been fully constructed. Then the subtrees rooted at the remaining children of the node are constructed in their left-to-right order. The state of the parser is defined by the current stack of nodes, the queue of remaining input words and the partial structure specified so far. The parser starts with an artificial *root* element in the stack and terminates when it reaches a configuration with an empty queue and with the artificial root element on the top of the stack. The algorithm uses 3 main types of decisions:

1. The decision $\mathbf{Shift}_w$ shifts the word w from the queue to the stack.
2. The decision $\mathbf{Project}_Y$ replaces the current top of the stack X with a new node Y, and specifies that Y is the parent of X in the output structure.
3. The decision **Attach** removes the current top of the stack X and specifies that element Y under the top of the stack is the parent of X.

Though these three types of decisions are sufficient to parse any constituent tree, Henderson (2003) extends the parsing strategy to include a specific treatment of a particular configuration in the parse tree, *Chomsky adjunction*, using a version of the Attach decision called **Modify**.

As was defined in expression (5), the probability of each state variable s_j^t in the ISBN depends on all the latent variables and previous relevant decisions in a subset of previous relevant positions t' : $r(t',t)$. In this ISBN model for phrase structure parsing, we use the same pattern of interconnections between variables as in the neural network of Henderson (2003), where there are different relations

$r(t',t)$ for selecting previous decision variables $D^{t'}$ and for selecting previous latent variables $S^{t'}$. Namely, the following four types of relations for selecting the previous positions $t' : r(t',t)$ for latent variables $S^{t'}$ are used:

1. *Stack Context*: the last previous position with the same element on top of the stack as at current position t.
2. *Sub-Top of Stack*: the last previous position where the node under the current top of the stack was on top of the stack.
3. *Left Child of Top of Stack*: the last previous position where the leftmost child of the current stack top was on top of the stack.
4. *Right Child of Top of Stack*: the last previous position where the rightmost child of the current stack top was on top of the stack.

These relations were motivated by linguistic considerations and many of them have also been found useful in other parsing models (Johnson, 1998; Roark and Johnson, 1999). Also, this set of relations ensures that the immediately preceding state is always included somewhere in the set of connected states. This requirement ensures that information, at least theoretically, can pass between any two states in the decision sequence, thereby avoiding any hard independence assumptions. Also note that each relation only selects at most one position (the most recent one of that kind). This ensures that the number of such connections to a latent vector remains bounded at four, so it should generalise well across larger, more complex constituency structures.

For selecting the previous positions $t' : r(t',t)$ for decision variables $D^{t'}$, the following relations are use:

1. *Previous*: the previous position $t-1$.
2. *Top*: the position at which the current top of the stack was shifted (if it is a terminal) or introduced (if non-terminal).
3. *Last Shift*: the position at which the last terminal was shifted.
4. *Left Terminal of Top of Stack*: the position when the leftmost terminal dominated by the current stack top was shifted.

This set includes the previous decision (*Previous*), which is important if the model does not do backward reasoning, as in the feed-forward approximation. The remaining relations pick out important labels, part-of-speech tags, and words in the context.

We used the Penn Treebank Wall Street Journal corpus to perform the empirical evaluation of the considered approaches. It is expensive to train the IMF approximation on the whole WSJ corpus, so instead we both trained and tested the model only on sentences of length at most 15, as in Taskar et al. (2004); Turian et al. (2006); Finkel et al. (2008). The standard split of the corpus into training (9,753 sentences, 104,187 words), validation (321 sentences, 3,381 words), and testing (603 sentences, 6,145 words) was performed.

As in Henderson (2003) and Turian and Melamed (2006) we used a publicly available tagger (Ratnaparkhi, 1996) to provide the part-of-speech tag for each word in the sentence. For each tag, there is an unknown-word vocabulary item which is used for all those words which are not sufficiently frequent with that tag to be included individually in the vocabulary. We only included a specific tag-word pair in the vocabulary if it occurred at least 20 time in the training set, which (with tag-unknown-word pairs) led to the very small vocabulary of 567 tag-word pairs.

	R	P	F_1
Bikel, 2004	87.9	88.8	88.3
Taskar et al., 2004	89.1	89.1	89.1
NN method	**89.1**	**89.2**	**89.1**
Turian et al., 2006	89.3	89.6	89.4
IMF method	**89.3**	**90.7**	**90.0**
Charniak, 2000	90.0	90.2	90.1
Finkel et al., 2008, 'feature-based'	91.1	90.2	90.6

Table 1: Percentage labelled constituent recall (R), precision (P), combination of both (F_1) on the testing set.

For decoding, we use best-first search with the search space pruned in two different ways. First, only a fixed number of the most probable partial derivations are pursued after each word shift operation. Secondly, the branching factor at each decision is limited. In the experiments presented in this chapter, we used the post-shift beam width of 10 and the branching factor of 5. Increasing the beam size and the branching factor beyond these values did not significantly effect parsing accuracy. For both of the models, the state vector length of 40 was used. All the parameters for both the NN and IMF models were tuned on the validation set. A single best model of each type was then applied to the final testing set.

Table 1 lists the results of the NN approximation and the IMF approximation,[7] along with results of different generative and discriminative parsing methods evaluated in the same experimental setup (Bikel, 2004; Taskar et al., 2004; Turian et al., 2006; Charniak, 2000; Finkel et al., 2008).[8] The IMF model improves over the baseline NN approximation, with a relative error reduction in F-measure exceeding 8%. This improvement is statistically significant.

The IMF model achieves results which do not appear to be significantly different from the results of the best model in the list (Charniak, 2000). Although no longer one of the most accurate parsing models on the standard WSJ parsing benchmark (including sentences of all lengths), the (Charniak, 2000) parser achieves competitive results (89.5% F-measure) and is still considered a viable approach, so the results reported here confirm the viability of our models. It should also be noted that previous results for the NN approximation to ISBNs on the standard WSJ benchmark (Henderson, 2003, 2004) achieved accuracies which are still competitive with the state of the art (89.1% F-measure for Henderson, 2003 and 90.1% F-measure for Henderson, 2004). For comparison, the LPCFG model of Petrov et al. (2006) achieve 89.7% F-measure on the standard WSJ benchmark.

We do not report the results on our data set of the LPCFG model of Petrov et al. (2006), probably the most relevant previous work on grammar learning (see the extended discussion in Section 8), as it would require tuning of their split-merge EM algorithm to achieve optimal results on the smaller

7. Approximate training times on a standard desktop PC for the IMF and NN approximations were 140 and 3 hours, respectively, and parsing times were 3 and 0.05 seconds per token, respectively. Parsing with the IMF method could be made more efficient, for example by not requiring the numerical approximations to reach convergence.

8. The results for the models of Bikel (2004) and Charniak (2000) trained and tested on sentences of length at most 15 were originally reported by Turian and Melamed (2005).

data set. However, we note that the CRF-based model of Finkel et al. (2008) (the reported 'feature-based' version) and the LPCFG achieves very close results when trained and tested on the sentences of length under 100 (Finkel et al., 2008) and, therefore, would be expected to demonstrate similar results in our setting. Note also that the LPCFG decoding algorithm uses a form of Bayes risk minimisation to optimise for the specific scoring function, whereas our model, as most parsing methods in the literature, output the highest scoring tree (maximum a-posteriori decoding). In fact, approximate Bayes risk minimisation can be used with our model and in our previous experiments resulted in approximately 0.5% boost in performance (Titov and Henderson, 2006). We chose not to use it here, as the maximum a-posteriori decoding is simpler, more widely accepted and, unlike Bayes risk minimisation, is expected to result in self-consistent trees.

These experimental results suggest that ISBNs are an appropriate model for structured prediction. Even approximations such as those tested here, with a very strong factorisability assumption, allow us to build quite accurate parsing models. We believe this provides strong justification for work on more accurate approximations of ISBNs.

8. Additional Related Work

Whereas graphical models are standard models for sequence processing, there has not been much previous work on graphical models for the prediction of structures more complex than sequences. Sigmoid belief networks were used originally for character recognition tasks, but later a Markovian dynamic extension of this model was applied to the reinforcement learning task (Sallans, 2002). However, their graphical model, approximation method, and learning method differ substantially from those of this paper.

When they were originally proposed, latent variable models for natural language parsing were not particularly successful, demonstrating results significantly below the state-of-the-art models (Kurihara and Sato, 2004; Matsuzaki et al., 2005; Savova and Peshkin, 2005; Riezler et al., 2002) or they were used in combination with already state-of-the-art models (Koo and Collins, 2005) and demonstrated a moderate improvement. More recently several methods (Petrov et al., 2006; Petrov and Klein, 2007; Liang et al., 2007), framed as grammar refinement approaches, demonstrated results similar to the best results achieved by generative models. All these approaches considered extensions of a classic PCFG model, which augment non-terminals of the grammar with latent variables (Latent-annotated PCFGs, LPCFGs). Even though marginalisation can be performed efficiently by using dynamic programming, decoding under this model is NP-hard (Matsuzaki et al., 2005; Sima'an, 1992). Instead, approximate parsing algorithms were considered.

The main reason for the improved performance of the more recent LPCFG methods is that they address the problem that with LPCFGs it is difficult to discover the appropriate latent variable augmentations for non-terminals. Early LPCFG models which used straight-forward implementations of expectation maximisation algorithms did not achieve state-of-the-art results (Matsuzaki et al., 2005; Prescher, 2005). To solve this problem the split-and-merge approach was considered in Petrov et al. (2006); Petrov and Klein (2007) and Dirichlet Process priors in Liang et al. (2007). The model of Petrov and Klein (2007) achieved the best reported result for a single model parser (90.1% F-measure). Even with the more sophisticated learning methods, in all of the work on LPCFGs the number of latent annotations which are successfully learnt is small, compared to the 40-dimensional vectors used in our experiments with ISBNs.

One important difference between LPCFGs and ISBNs is that in LPCFGs the latent annotations are used to expand the set of atomic labels used in a PCFG, whereas ISBNs directly reason with a vector of latent features. This use of a vector space instead of atomic labels provides ISBNs with a much larger label space with a much richer structure of similarity between labels, based on shared features. This highly structured label space allows standard gradient descent techniques to work well even with large numbers of latent features. In contrast, learning for LPCFGs has required the specialised methods discussed above and has succeeded in searching a much more limited space of latent annotations. These specialised methods impose a hierarchical structure of similarity on the atomic labels of LPCFGs, based on recursive binary augmentations of labels ("splits"), but this hierarchical structure is much less rich that the similarity structure of a vector space.

Another important difference with LPCFGs is that ISBN models do not place strong restrictions on the structure of statistical dependencies between latent variables, such as the context-free restriction of LPCFGs. This makes ISBNs easily applicable to a much wider set of problems. For example, ISBNs have been applied to the dependency parsing problem (Titov and Henderson, 2007) and to joint dependency parsing and semantic role labelling (Henderson et al., 2008; Gesmundo et al., 2009), where in both cases they achieved state-of-the-art results. The application of LPCFG models to even dependency parsing has required sophisticated grammar transformations (Musillo and Merlo, 2008), to which the split-and-merge training approach has not yet been successfully adapted.

The experiments reported in Henderson et al. (2008) also suggest that the latent annotations of syntactic states are not only useful for syntactic parsing itself but also can be helpful for other tasks. In these experiments, semantic role labelling performance rose by about 3.5% when latent annotations for syntactic decision were provided, thereby indicating that the latent annotation of syntactic parsing states helps semantic role labelling.

9. Conclusions

This paper proposes a new class of graphical models for structured prediction problems, incremental sigmoid belief networks, and has applied it to natural language grammar learning. ISBNs allow the structure of the graphical model to be dependent on the output structure. This allows the model to directly express regularities that are local in the output structure but not local in the input structure, making ISBNs appropriate for parsing problems. This ability supports the induction of latent variables which augment the grammatical structures annotated in the training data, thereby solving a limited form of grammar induction. Exact inference with ISBNs is not tractable, but we derive two tractable approximations. First, it is shown that the feed-forward neural network of Henderson (2003) can be considered as a simple approximation to ISBNs. Second, a more accurate but still tractable approximation based on mean field theory is proposed.

Both approximation models are empirically evaluated. First, artificial experiments were performed, where both approximations significantly outperformed a baseline. The mean field method achieved average relative error reduction of about 27% over the neural network approximation, demonstrating that it is a more accurate approximation. Second, both approximations were applied to the natural language parsing task, where the mean field method demonstrated significantly better results. These results are non-significantly different from the results of another history-based probabilistic model of parsing (Charniak, 2000) which is competitive with the state-of-the-art for single-model parsers. The fact that a more accurate approximation leads to a more accurate parser

suggests that the ISBNs proposed here are a good abstract model for grammar learning. This empirical result motivates further research into more accurate approximations of ISBNs.

Acknowledgments

We would like to acknowledge support for this work from the Swiss NSF scholarship PBGE22-119276 and from the European Commission (EC) funded CLASSiC project (project number 216594 funded under EC FP7, Call 1), and the partial support of the Excellence Cluster on Multimodal Computing and Interaction (MMCI) at Saarland University. We would also like to thank Paola Merlo for extensive discussions and Mark Johnson for his helpful comments on this work.

Appendix A.

This appendix presents details of computing gradients for the incremental mean field approximation. We perform maximum likelihood estimation of the ISBN parameters, using the estimator of the structure probability defined in expression (8). We focus on the incremental mean field approximation introduced in Section 5.3. As we have shown there, estimates of the conditional distribution $q_k^t(d) \approx P(d_k^t = d|h(t,k))$ are dependent on the means $\mu^{t,k}$ computed at the elementary step (t,k) in the same way as the estimates $q_k^t(d)$ in the feed-forward approximation depend on the means μ^t in expression (10), that is,

$$q_k^t(d) = \frac{\Phi_{h(t,k)}(d)\exp(\sum_j W_{dj}\mu_j^{t,k})}{\sum_{d'}\Phi_{h(t,k)}(d')\exp(\sum_j W_{d'j}\mu_j^{t,k})}. \tag{15}$$

We use the gradient descent algorithm, so the goal of this section is to describe how to compute derivatives of the log-likelihood

$$\hat{L}(T) = \sum_{t,k}\sum_j W_{d_k^t j}\mu_j^{t,k} - \log\left(\sum_{d'}\Phi_{h(t,k)}(d')\exp(\sum_j W_{d'j}\mu_j^{t,k})\right)$$

with respect to all the model parameters. The derivatives of $\hat{L}(T)$ with respect to model parameters can be expressed as

$$\frac{d\hat{L}(T)}{dx} = \sum_{d,j}\frac{\partial\hat{L}(T)}{\partial W_{dj}}\frac{dW_{dj}}{dx} + \sum_{t,k,i}\frac{\partial\hat{L}(T)}{\partial\mu_i^{t,k}}\frac{d\mu_i^{t,k}}{dx}, \tag{16}$$

where x is any model parameter, that is, entries of the weight matrices J, B and W. All the terms except for $\frac{d\mu_i^{t,k}}{dx}$ are trivial to compute:

$$\frac{\partial\hat{L}(T)}{\partial W_{dj}} = \sum_{t,k}\mu_j^{t,k}\left(\delta_{d_k^t d} - q_k^t(d)\right), \tag{17}$$

$$\frac{\partial\hat{L}(T)}{\partial\mu_i^{t,k}} = (1-\delta_{k,K_t+1})\left(W_{d_k^t i} - \sum_d q_k^t(d)W_{di}\right),$$

where δ_{ij} is the Kronecker delta. Computation of the total derivatives $\frac{d\mu_i^{t,k}}{dx}$ is less straightforward. The main challenge is that dependence of $\mu_j^{t,k}$ for $k > 1$ on other model parameters cannot be expressed analytically, as we found values of $\mu_j^{t,k}$ by performing numerical maximisation of the expression $L_V^{t,k}$ (13). In the next several paragraphs we will consider only the case of $k > 1$, but later we will return to the simpler case of $k = 1$, where the computation of derivatives is equivalent to the backpropagation algorithm in standard feed-forward neural networks.

Note that the gradient of the log-likelihood can be easily computed in the standard mean field methods for SBNs (Saul and Jordan, 1999; Saul et al., 1996), even though they also use numeric strategies to find optimal means. There means are selected so as to maximise the variational upper bound L_V (4), which is used as the log-likelihood $\hat{L} = L_V$ in their approach. In static SBNs it is feasible to perform complete maximisation of the entire $\hat{L}$, which involves multiple backward-forward passes through the structure of the graphical model. This leads to all the derivatives $\frac{d\hat{L}}{d\mu_i}$ being equal to zero. Therefore, no error backpropagation is needed in their case. All the derivatives $\frac{d\hat{L}}{dx}$ can be computed using variational parameters associated with the nodes corresponding to the parameter x. E.g. if x is a weight of an edge then only variational parameters associated with the variables at its ends are needed to compute the derivative. Unfortunately, learning with the incremental mean field approximation proposed in this paper is somewhat more complex.

In order to compute derivatives $\frac{d\mu_i^{t,k}}{dx}$ we assume that maximisation of $L_V^{t,k}$ is done until convergence, then the partial derivatives of $L_V^{t,k}$ with respect to $\mu_i^{t,k}$ are equal to zero. This gives us a system of linear equations, which describes interdependencies between the current means $\mu^{t,k}$, the previous means $\mu^{t,k-1}$ and the weights W:

$$F_i^{t,k} = \frac{\partial L_V^{t,k}}{\partial \mu_i^{t,k}} = \ln(1-\mu_i^{t,k}) - \ln\mu_i^{t,k} - \ln(1-\mu_i^{t,k-1}) + \ln\mu_i^{t,k-1} + W_{d_{k-1}^t i} - \sum_d \hat{q}_{k-1}^t(d) W_{di} = 0,$$

for $1 < i \leq n$, where $\hat{q}_{k-1}^t$ is the distribution over decisions computed in the same way as q_{k-1}^t (15), but using means $\mu_i^{t,k}$ instead of $\mu_i^{t,k-1}$:

$$\hat{q}_{k-1}^t(d) = \frac{\Phi_{h(t,k-1)}(d) \exp(\sum_j W_{dj} \mu_j^{t,k})}{\sum_{d'} \Phi_{h(t,k-1)}(d') \exp(\sum_j W_{d'j} \mu_j^{t,k})}.$$

This system of equations permits the use of implicit differentiation to compute the derivatives $\frac{\partial \mu_i^{t,k}}{\partial z}$, where z can be a weight matrix component W_{dj} or a previous mean $\mu_j^{t,k-1}$ involved in expression (13). It is important to distinguish z from x, used above, because x can be an arbitrary model parameter not necessary involved in the expression $L_V^{t,k}$ but affecting the current means $\mu_i^{t,k}$ through $\mu_j^{t,k-1}$. Equally important to distinguish partial derivatives $\frac{\partial \mu_i^{t,k}}{\partial z}$ from the total derivatives $\frac{d\mu_i^t}{dz}$, because the dependency of $\mu_i^{t,k}$ on parameter z can be both direct, through maximisation of $L_V^{t,k}$, but also indirect through previous maximisation steps (t',k'), where $L_V^{t',k'}$ was dependent on z. The relation between the total and partial derivatives can be expressed as

$$\frac{d\mu_i^{t,k}}{dz} = \frac{\partial \mu_i^{t,k}}{\partial z} + \sum_j \frac{\partial \mu_i^{t,k}}{\partial \mu_j^{t,k-1}} \frac{d\mu_j^{t,k-1}}{dz},$$

meaning that indirect dependencies of $\mu_i^{t,k}$ ($k > 1$) on parameters z are coming through previous means $\mu_j^{t,k-1}$. We apply the implicit differentiation theorem and obtain the vector of partial derivatives with respect to a parameter z $D_z\mu^t = \{\frac{\partial \mu_1^t}{\partial z}, \ldots, \frac{\partial \mu_n^t}{\partial z}\}$ as

$$D_z\mu^t = -\left(D_{\mu^t}F^{t,k}\right)^{-1} D_z F^{t,k}, \tag{18}$$

where $D_{\mu^t}F^{t,k}$ and $D_z F^{t,k}$ are Jacobians:

$$D_{\mu^t}F^{t,k} = \begin{pmatrix} \frac{\partial F_1^{t,k}}{\partial \mu_1^t} & \cdots & \frac{\partial F_1^{t,k}}{\partial \mu_n^t} \\ \cdots & \cdots & \cdots \\ \frac{\partial F_n^{t,k}}{\partial \mu_1^t} & \cdots & \frac{\partial F_n^{t,k}}{\partial \mu_n^t} \end{pmatrix}, \quad D_z F^{t,k} = \begin{pmatrix} \frac{\partial F_1^{t,k}}{\partial z} \\ \cdots \\ \frac{\partial F_n^{t,k}}{\partial z} \end{pmatrix}.$$

Now we derive the Jacobians $D_{\mu^t}F^{t,k}$ and $D_z F^{t,k}$ for different types of parameters z. The matrix $D_{\mu^t}F^{t,k}$ consists of the components

$$\begin{aligned} \frac{\partial F_i^{t,k}}{\partial \mu_j^{t,k}} = &- \frac{\delta_{ij}}{\mu_j^{t,k}(1-\mu_j^{t,k})} - \sum_d \hat{q}_{k-1}^t(d) W_{di} W_{dj} \\ &+ \left(\sum_d \hat{q}_{k-1}^t(d) W_{di}\right)\left(\sum_d \hat{q}_{k-1}^t(d) W_{dj}\right), \end{aligned} \tag{19}$$

where δ_{ij} is the Kronecker delta. If we consider $W_{\cdot i}$ as a random variable accepting values W_{di} under distribution $\hat{q}_{k-1}^t$, we can rewrite the Jacobian $D_{\mu^t}F^{t,k}$ as the negated sum of a positive diagonal matrix and the covariance matrix $\Sigma_{\hat{q}_{k-1}^t}(W)$. Therefore the matrix $D_{\mu^t}F^{t,k}$ is negative semidefinite.

Note that this matrix is the Hessian for the expression $L_V^{t,k}$ (13), which implies concavity of $L_V^{t,k}$ stated previously without proof. Similarly, the Hessian for (11) is only different by including output weight covariances for all the previous elementary decision, not only for the last one, and therefore expression (11) is also concave.

To conclude with the computation of $\frac{\partial \mu_i^{t,k}}{\partial z}$, we compute $D_{\mu^{t,k-1}}F^{t,k}$ and $D_W F^{t,k}$:

$$\frac{\partial F_i^{t,k}}{\partial \mu_j^{t,k-1}} = \frac{\delta_{ij}}{\mu_j^{t,k-1}(1-\mu_j^{t,k-1})}, \tag{20}$$

$$\frac{\partial F_i^{t,k}}{\partial W_{dj}} = \delta_{ij}\delta_{dd_k^t} - \hat{q}_{k-1}^t(d)\left(\delta_{ij} + (W_{di} - \sum_{d'} \hat{q}_{k-1}^t(d') W_{d'i})\mu_j^{t,k}\right). \tag{21}$$

Now the partial derivatives $\frac{\partial \mu_i^{t,k}}{\partial W_{dj}}$ and $\frac{\partial \mu_i^{t,k}}{\partial \mu_i^{t,k-1}}$ can be computed by substituting expressions (19)-(21) into (18).

For $k = 1, \mu_i^{t,1}$ was shown to be equal to the sigmoid function of the weighted sum of the parents means as defined in (12) and (14). Therefore, we can compute the partial derivatives of $\mu_i^{t,1}$ with respect to other means and parameters involved in (12) and (14):

$$\frac{\partial \mu_i^{t,1}}{\partial \mu_j^{t',K_{t'}+1}} = \sigma'(\eta_i^t) \sum_{r:r(t',t)} J_{ij}^r,$$

$$\frac{\partial \mu_i^{t,1}}{\partial J_{jl}^r} = \delta_{ij}\sigma'(\eta_i^t) \sum_{t':r(t',t)} \mu_l^{t',K_{t'}+1},$$

$$\frac{\partial \mu_i^{t,1}}{\partial B_{jd}^{rk}} = \delta_{ij}\sigma'(\eta_i^t) \sum_{t':r(t',t)} \delta_{dd_k^{t'}}.$$

where $\sigma'(\eta_i^t) = \sigma(\eta_i^t)\,(1-\sigma(\eta_i^t))$.

In order to compute $\frac{d\mu_i^{t,k}}{dx}$ in (16), derivatives with respect to previous means $\frac{\partial \mu_i^{t,k}}{\partial \mu_i^{t,k-1}}$ are used to propagate the error in a similar way to the neural network backpropagation algorithm (Rumelhart et al., 1986). We denote the total derivative of the approximate log-likelihood with respect to the means of the latent variables as $\varepsilon_i^{t,k} = \frac{d\hat{L}(T)}{d\mu_i^{t,k}}$. The incrementality of the mean field algorithm guarantees that latent vectors of means $\mu^{t,k}$ are computed from the means of the previous elementary steps. Therefore, values $\varepsilon_i^{t,k}$ can be computed in the opposite order, propagating the information back through the structure. Namely, the recursive formulae would be:

$$\varepsilon_i^{t,k} = \frac{\partial \log q_k^t}{\partial \mu_i^{t,k}} + \sum_j \varepsilon_j^{t,k+1} \frac{\partial \mu_j^{t,k+1}}{\partial \mu_i^{t,k}}, \;\; k \le K_t \;,$$

$$\varepsilon_i^{t,K^t+1} = \sum_{r,t':r(t,t')} \sum_j \varepsilon_j^{t',1} \sigma'(\eta_j^{t'}) J_{ji}^r.$$

After computing values ε for all the elementary steps (t,k), we can evaluate the derivatives of the model parameters. We start with the output distribution parameters W_{di}:

$$\frac{d\hat{L}(T)}{dW_{di}} = \frac{\partial \hat{L}(T)}{\partial W_{di}} + \sum_{t,k} \sum_j \varepsilon^{t,k} \frac{\partial \mu_j^{t,k}}{\partial W_{di}}.$$

The first term here is evaluated as defined in (17), the term $\frac{\partial \mu_j^{t,k}}{\partial W_{di}}$ is computed as explained above.

Finally, the total derivatives of the log-likelihood with respect to the parameters J_{ij}^t and B_{id}^{rk} are found as follows

$$\frac{d\hat{L}(T)}{dJ_{ij}^r} = \sum_t \mu_j^t \sum_{t':r(t,t')} \varepsilon_i^{t',1} \sigma'(\eta_i^{t'}),$$

$$\frac{d\hat{L}(T)}{dB_{id}^{rk}} = \sum_t \delta_{d_k^t d} \sum_{t':r(t,t')} \varepsilon_i^{t',1} \sigma'(\eta_i^{t'}).$$

References

Dan M. Bikel. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511, 2004.

Leon Bottou. *Une approche théoretique de l'apprentissage connexionniste: Applications à la reconnaissance de la parole*. PhD thesis, Université de Paris XI, Paris, France, 1991.

Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of North American Chapter of Association for Computational Linguistics*, pages 132–139, Seattle, Washington, 2000.

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Meeting of Association for Computational Linguistics*, pages 173–180, Ann Arbor, MI, 2005.

Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1999.

Persi Diaconis and Bradley Efron. Computer-intensive methods in statistics. *Scientific American*, pages 116–130, 1983.

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK, 1998.

Jenny R. Finkel, Alex Kleeman, and Christopher D. Manning. Efficient feature-based, conditional random field parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 959–967, Columbus, Ohio, June 2008.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. A latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning: Shared Task*, pages 37–42, 2009.

James Henderson. Inducing history representations for broad coverage statistical parsing. In *Proceedings of the joint meeting of North American Chapter of the Association for Computational Linguistics and the Human Language Technology Conference*, pages 103–110, Edmonton, Canada, 2003.

James Henderson. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Meeting of Association for Computational Linguistics*, Barcelona, Spain, 2004.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning*, pages 178–182, Manchester, UK, 2008.

Geoffrey E. Hinton, Peter Dayan, Brendan Frey, and Radford Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995.

Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4): 613–632, 1998.

Michael I. Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. In Michael I. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

Terry Koo and Michael Collins. Hidden-variable models for discriminative reranking. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 507–514, 2005.

Kenichi Kurihara and Taisuke Sato. An application of the variational bayesian approach to probabilistic context-free grammars. In *Proceedings of the International Joint Conference on Natural Language Processing, Workshop: Beyond Shallow Analyses*, 2004.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. The infinite PCFG using hierarchical dirichlet processes. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 688–697, 2007.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 75–82, 2005.

Kevin P. Murphy. *Dynamic Belief Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, CA, 2002.

Gabriele Musillo and Paola Merlo. Unlexicalised hidden variable models of split dependency grammars. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, 2008.

Radford Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of the Conference on Human Language Technology and North American chapter of the Association for Computational Linguistics*, pages 404–411, 2007.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, 2006.

Detlef Prescher. Head-driven PCFGs with latent-head statistics. In *Proceedings of the 9th International Workshop on Parsing Technology*, pages 115–124, 2005.

William H. Press, Brian Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes*. Cambridge University Press, Cambridge, UK, 1996.

Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Univ. of Pennsylvania, PA, 1996.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, III, and Mark Johnson. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 271–278, 2002.

Brian Roark and Mark Johnson. Efficient probabilistic top-down and left-corner parsing. In *Proceedings of the 37th Meeting of Association for Computational Linguistics*, pages 421–428, 1999.

Khashayar Rohanimanesh, Michael Wick, and Andrew McCallum. Inference and learning in large factor graphs with adaptive proposal distributions and a rank-based objective. Technical Report UM-CS-2009-008, University of Massachusetts, 2009.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing, Vol 1*, pages 318–362. MIT Press, Cambridge, MA, 1986.

Brian Sallans. *Reinforcement Learning for Factored Markov Decision Processes*. PhD thesis, University of Toronto, Toronto, Canada, 2002.

Lawrence K. Saul and Michael I. Jordan. A mean field learning algorithm for unsupervised neural networks. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 541–554. MIT Press, Cambridge, MA, 1999.

Lawrence K. Saul, Tommi Jaakkola, and Michael I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

Virginia Savova and Leon Peshkin. Dependency parsing with dynamic Bayesian network. In *AAAI, 20th National Conference on Artificial Intelligence*, pages 1112–1117, Pittsburgh, Pennsylvania, 2005.

Khalil Sima'an. Computational complexity of probabilistic disambiguation. *Grammars*, 5:125–151, 1992.

Eljas Soisalon-Soininen and Esko Ukkonen. A method for transforming grammars into LL(k) form. *Acta Informatica*, 12:339–369, 1979.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher D. Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain, 2004.

Ivan Titov and James Henderson. Loss minimization in parse reranking. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 560–567, 2006.

Ivan Titov and James Henderson. A latent variable model for generative dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 144–155, 2007.

Joseph Turian and I. Dan Melamed. Constituent parsing by classification. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 141–151, 2005.

Joseph Turian and I. Dan Melamed. Advances in discriminative parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Conference on Computational Linguistics*, Sydney, Australia, 2006.

Joseph Turian, Benjamin Wellington, and I. Dan Melamed. Scalable discriminative learning for natural language parsing and translation. In *Proc. 20th Conference on Neural Information Processing Systems*, Vancouver, Canada, 2006.

Journal of Machine Learning Research 11 (2010) 3571-3594 Submitted 4/10; Revised 10/10; Published 12/10

Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory

Sumio Watanabe SWATANAB@PI.TITECH.AC.JP
Precision and Intelligence Laboratory
Tokyo Institute of Technology
Mailbox R2-5, 4259 Nagatsuta, Midori-ku
Yokohama 226-8503, Japan

Editor: Manfred Opper

Abstract

In regular statistical models, the leave-one-out cross-validation is asymptotically equivalent to the Akaike information criterion. However, since many learning machines are singular statistical models, the asymptotic behavior of the cross-validation remains unknown. In previous studies, we established the singular learning theory and proposed a widely applicable information criterion, the expectation value of which is asymptotically equal to the average Bayes generalization loss. In the present paper, we theoretically compare the Bayes cross-validation loss and the widely applicable information criterion and prove two theorems. First, the Bayes cross-validation loss is asymptotically equivalent to the widely applicable information criterion as a random variable. Therefore, model selection and hyperparameter optimization using these two values are asymptotically equivalent. Second, the sum of the Bayes generalization error and the Bayes cross-validation error is asymptotically equal to $2\lambda/n$, where λ is the real log canonical threshold and n is the number of training samples. Therefore the relation between the cross-validation error and the generalization error is determined by the algebraic geometrical structure of a learning machine. We also clarify that the deviance information criteria are different from the Bayes cross-validation and the widely applicable information criterion.

Keywords: cross-validation, information criterion, singular learning machine, birational invariant

1. Introduction

A statistical model or a learning machine is said to be regular if the map taking parameters to probability distributions is one-to-one and if its Fisher information matrix is positive definite. If a model is not regular, then it is said to be singular. Many learning machines, such as artificial neural networks (Watanabe, 2001b), normal mixtures (Yamazaki and Watanabe, 2003), reduced rank regressions (Aoyagi and Watanabe, 2005), Bayes networks (Rusakov and Geiger, 2005; Zwiernik, 2010), mixtures of probability distributions (Lin, 2010), Boltzmann machines (Aoyagi, 2010), and hidden Markov models (Yamazaki and Watanabe, 2005), are not regular but singular (Watanabe, 2007). If a statistical model or a learning machine contains a hierarchical structure, hidden variables, or a grammatical rule, then the model is generally singular. Therefore, singular learning theory is necessary in modern information science.

The statistical properties of singular models have remained unknown until recently, because analyzing a singular likelihood function had been difficult (Hartigan, 1985; Watanabe, 1995). In singular statistical models, the maximum likelihood estimator does not satisfy asymptotic normality.

Consequently, AIC is not equal to the average generalization error (Hagiwara, 2002), and the Bayes information criterion (BIC) is not equal to the Bayes marginal likelihood (Watanabe, 2001a), even asymptotically. In singular models, the maximum likelihood estimator often diverges, or even if it does not diverge, makes the generalization error very large. Therefore, the maximum likelihood method is not appropriate for singular models. On the other hand, Bayes estimation was proven to make the generalization error smaller if the statistical model contains singularities. Therefore, in the present paper, we investigate methods for estimating the Bayes generalization error.

Recently, new statistical learning theory, based on methods from algebraic geometry, has been established (Watanabe, 2001a; Drton et al., 2009; Watanabe, 2009, 2010a,c; Lin, 2010). In singular learning theory, a log likelihood function can be made into a common standard form, even if it contains singularities, by using the resolution theorem in algebraic geometry. As a result, the asymptotic behavior of the posterior distribution is clarified, and the concepts of BIC and AIC can be generalized onto singular statistical models. The asymptotic Bayes marginal likelihood was proven to be determined by the real log canonical threshold (Watanabe, 2001a), and the average Bayes generalization error was proven to be estimable by the widely applicable information criterion (Watanabe, 2009, 2010a,c).

Cross-validation is an alternative method for estimating the generalization error (Mosier, 1951; Stone, 1977; Geisser, 1975). By definition, the average of the cross-validation is equal to the average generalization error in both regular and singular models. In regular statistical models, the leave-one-out cross-validation is asymptotically equivalent to AIC (Akaike, 1974) in the maximum likelihood method (Stone, 1977; Linhart, 1986; Browne, 2000). However, the asymptotic behavior of the cross-validation in singular models has not been clarified.

In the present paper, in singular statistical models, we theoretically compare the Bayes cross-validation, the widely applicable information criterion, and the Bayes generalization error and prove two theorems. First, we show that the Bayes cross-validation loss is asymptotically equivalent to the widely applicable information criterion as a random variable. Second, we also show that the sum of the Bayes cross-validation error and the Bayes generalization error is asymptotically equal to $2\lambda/n$, where λ is the real log canonical threshold and n is the number of training samples. It is important that neither λ or n is a random variable. Since the real log canonical threshold is a birational invariant of the statistical model, the relationship between the Bayes cross-validation and the Bayes generalization error is determined by the algebraic geometrical structure of the statistical model.

The remainder of the present paper is organized as follows. In Section 2, we introduce the framework of Bayes learning and explain singular learning theory. In Section 3, the Bayes cross-validation is defined. In Section 4, the main theorems are proven. In Section 5, we discuss the results of the present paper, and the differences among the cross-validation, the widely applicable information criterion, and the deviance information criterion are investigated theoretically and experimentally. Finally, in Section 6, we summarize the primary conclusions of the present paper.

2. Bayes Learning Theory

In this section, we summarize Bayes learning theory for singular learning machines. The results presented in this section are well known and are the fundamental basis of the present paper. Table 1 lists variables, names, and equation numbers in the present paper.

Variable	Name	Equation number
$\mathbb{E}_w[\ ]$	posterior average	Equation (1)
$\mathbb{E}_w^{(i)}[\ ]$	posterior average without X_i	Equation (16)
$L(w)$	log loss function	Equation (8)
L_0	minimum loss	Equation (9)
L_n	empirical loss	Equation (10)
$B_gL(n)$	Bayes generalization loss	Equation (3)
$B_tL(n)$	Bayes training loss	Equation (4)
$G_tL(n)$	Gibbs training loss	Equation (7)
$C_vL(n)$	cross-validation loss	Equation (17)
$B_g(n)$	Bayes generalization error	Equation (11)
$B_t(n)$	Bayes training error	Equation (12)
$C_v(n)$	cross-validation error	Equation (28)
$V(n)$	functional variance	Equation (5)
$Y_k(n)$	kth functional cumulant	Equation (18)
WAIC(n)	WAIC	Equation (6)
λ	real log canonical threshold	Equation (29)
ν	singular fluctuation	Equation (30)

Table 1: Variables, Names, and Equation Numbers

2.1 Framework of Bayes Learning

First, we explain the framework of Bayes learning.

Let $q(x)$ be a probability density function on the N dimensional real Euclidean space $\mathbb{R}^N$. The training samples and the testing sample are denoted by random variables $X_1, X_2, ..., X_n$ and X, respectively, which are independently subject to the same probability distribution as $q(x)dx$. The probability distribution $q(x)dx$ is sometimes called the true distribution.

A statistical model or a learning machine is defined as a probability density function $p(x|w)$ of $x \in \mathbb{R}^N$ for a given parameter $w \in W \subset \mathbb{R}^d$, where W is the set of all parameters. In Bayes estimation, we prepare a probability density function $\varphi(w)$ on W. Although $\varphi(w)$ is referred to as a prior distribution, in general, $\varphi(w)$ does not necessary represent an *a priori* knowledge of the parameter.

For a given function $f(w)$ on W, the expectation value of $f(w)$ with respect to the posterior distribution is defined as

$$\mathbb{E}_w[f(w)] = \frac{\displaystyle\int f(w) \prod_{i=1}^{n} p(X_i|w)^{\beta}\, \varphi(w) dw}{\displaystyle\int \prod_{i=1}^{n} p(X_i|w)^{\beta}\, \varphi(w) dw}, \tag{1}$$

where $0 < \beta < \infty$ is the inverse temperature. The case in which $\beta = 1$ is most important because this case corresponds to strict Bayes estimation. The Bayes predictive distribution is defined as

$$p^*(x) \equiv \mathbb{E}_w[p(x|w)]. \tag{2}$$

In Bayes learning theory, the following random variables are important. The Bayes generalization loss $B_gL(n)$ and the Bayes training loss $B_tL(n)$ are defined, respectively, as

$$B_gL(n) = -\mathbb{E}_X[\log p^*(X)], \tag{3}$$

$$B_tL(n) = -\frac{1}{n}\sum_{i=1}^{n}\log p^*(X_i), \tag{4}$$

where $\mathbb{E}_X[\ ]$ gives the expectation value over X. The *functional variance* is defined as

$$V(n) = \sum_{i=1}^{n}\Big\{\mathbb{E}_w[(\log p(X_i|w))^2] - \mathbb{E}_w[\log p(X_i|w)]^2\Big\}, \tag{5}$$

which shows the fluctuation of the posterior distribution. In previous papers (Watanabe, 2009, 2010a,b), we defined the widely applicable information criterion

$$\mathrm{WAIC}(n) \equiv B_tL(n) + \frac{\beta}{n}V(n), \tag{6}$$

and proved that

$$\mathbb{E}[B_gL(n)] = \mathbb{E}[\mathrm{WAIC}(n)] + o(\frac{1}{n}),$$

holds for both regular and singular statistical models, where $\mathbb{E}[\ ]$ gives the expectation value over the sets of training samples.

Remark 1 *Although the case in which* $\beta = 1$ *is most important, general cases in which* $0 < \beta < \infty$ *are also important for four reasons. First, from a theoretical viewpoint, several mathematical relations can be obtained using the derivative of* β*. For example, using the Bayes free energy or the Bayes stochastic complexity,*

$$\mathcal{F}(\beta) = -\log\int\prod_{i=1}^{n}p(X_i|w)^{\beta}\varphi(w)dw,$$

the Gibbs training loss

$$G_tL(n) = -\mathbb{E}_w\Big[\frac{1}{n}\sum_{i=1}^{n}\log p(X_i|w)\Big] \tag{7}$$

can be written as

$$G_tL(n) = \frac{\partial\mathcal{F}}{\partial\beta}.$$

Such relations are useful in investigating Bayes learning theory. We use $\partial^2\mathcal{F}/\partial\beta^2$ *to investigate the deviance information criteria in Section 5. Second, the maximum likelihood method formally corresponds to* $\beta = \infty$*. The maximum likelihood method is defined as*

$$p^*(x) = p(x|\hat{w}),$$

instead of Equation (2), where $\hat{w}$ is the maximum likelihood estimator. Its generalization loss is also defined in the same manner as Equation (3). In regular statistical models, the asymptotic Bayes generalization error does not depend on $0 < \beta \leq \infty$, whereas in singular models it strongly depends on β. Therefore, the general case is useful for investigating the difference between the maximum likelihood and Bayes methods. Third, from an experimental viewpoint, in order to approximate the posterior distribution, the Markov chain Monte Carlo method is often applied by controlling β. In particular, the identity

$$\mathcal{F}(1) = \int_0^1 \frac{\partial F}{\partial \beta}\, d\beta$$

is used in the calculation of the Bayes marginal likelihood. The theoretical results for general β are useful for monitoring the effect of controlling β (Nagata and Watanabe, 2008). Finally, in the regression problem, β can be understood as the variance of the unknown additional noise (Watanabe, 2010c) and so may be optimized as the hyperparameter. For these reasons, in the present paper, we theoretically investigate the cases for general β.

2.2 Notation

In the following, we explain the notation used in the present study.

The log loss function $L(w)$ and the entropy S of the true distribution are defined, respectively, as

$$\begin{aligned} L(w) &= -\mathbb{E}_X[\log p(X|w)], \\ S &= -\mathbb{E}_X[\log q(X)]. \end{aligned} \tag{8}$$

Then, $L(w) = S + D(q||p_w)$, where $D(q||p_w)$ is the Kullback-Leibler distance defined as

$$D(q||p_w) = \int q(x) \log \frac{q(x)}{p(x|w)} dx.$$

Then, $D(q||p_w) \geq 0$, hence $L(w) \geq S$. Moreover, $L(w) = S$ if and only if $p(x|w) = q(x)$.

In the present paper, we assume that there exists a parameter $w_0 \in W$ that minimizes $L(w)$,

$$L(w_0) = \min_{w \in W} L(w).$$

Note that such w_0 is not unique in general because the map $w \mapsto p(x|w)$ is, in general, not a one-to-one map in singular learning machines. In addition, we assume that, for an arbitrary w that satisfies $L(w) = L(w_0)$, $p(x|w)$ is the same probability density function. Let $p_0(x)$ be such a unique probability density function. In general, the set

$$W_0 = \{w \in W; p(x|w) = p_0(x)\}$$

is not a set of a single element but rather an analytic or algebraic set with singularities. Here, a set in $\mathbb{R}^d$ is said to be an analytic or algebraic set if and only if the set is equal to the set of all zero points of an analytic or algebraic function, respectively. For simple notations, the minimum log loss L_0 and the empirical log loss L_n are defined, respectively, as

$$L_0 = -\mathbb{E}_X[\log p_0(X)], \tag{9}$$

$$L_n = -\frac{1}{n}\sum_{i=1}^{n} \log p_0(X_i). \tag{10}$$

Then, by definition, $L_0 = \mathbb{E}[L_n]$. Using these values, Bayes generalization error $B_g(n)$ and Bayes training error $B_t(n)$ are defined, respectively, as

$$B_g(n) = B_g L(n) - L_0, \tag{11}$$
$$B_t(n) = B_t L(n) - L_n. \tag{12}$$

Let us define a log density ratio function as:

$$f(x,w) = \log \frac{p_0(x)}{p(x|w)},$$

which is equivalent to

$$p(x|w) = p_0(x)\exp(-f(x,w)).$$

Then, it immediately follows that

$$\begin{aligned} B_g(n) &= -\mathbb{E}_X[\log \mathbb{E}_w[\exp(-f(X,w))]], \\ B_t(n) &= -\frac{1}{n}\sum_{i=1}^{n} \log \mathbb{E}_w[\exp(-f(X_i,w))], \\ V(n) &= \sum_{i=1}^{n} \Big\{ \mathbb{E}_w[f(X_i,w)^2] - \mathbb{E}_w[f(X_i,w)]^2 \Big\}. \end{aligned}$$

Therefore, the problem of statistical learning is characterized by the function $f(x,w)$.

Definition 2 *(1) If $q(x) = p_0(x)$, then $q(x)$ is said to be realizable by $p(x|w)$. Otherwise, $q(x)$ is said to be unrealizable.*
(2) If the set W_0 consists of a single point w_0 and if the Hessian matrix $\nabla\nabla L(w_0)$ is strictly positive definite, then $q(x)$ is said to be regular for $p(x|w)$. Otherwise, $q(x)$ is said to be singular for $p(x|w)$.

Bayes learning theory was investigated for a realizable and regular case (Schwarz, 1978; Levin et al., 1990; Amari, 1993). The WAIC was found for a realizable and singular case (Watanabe, 2001a, 2009, 2010a) and for an unrealizable and regular case (Watanabe, 2010b). In addition, WAIC was generalized for an unrealizable and singular case (Watanabe, 2010d).

2.3 Singular Learning Theory

We summarize singular learning theory. In the present paper, we assume the followings.

2.3.1 ASSUMPTIONS

(1) The set of parameters W is a compact set in $\mathbb{R}^d$, the open kernel[1] of which is not the empty set. The boundary of W is defined by several analytic functions,

$$W = \{w \in \mathbb{R}^d; \pi_1(w) \geq 0, \pi_2(w) \geq 0, ..., \pi_k(w) \geq 0\}.$$

1. The open kernel of a set A is the largest open set that is contained in A.

(2) The prior distribution satisfies $\varphi(w) = \varphi_1(w)\varphi_2(w)$, where $\varphi_1(w) \geq 0$ is an analytic function and $\varphi_2(w) > 0$ is a C^∞-class function.
(3) Let $s \geq 8$ and let

$$L^s(q) = \{f(x); \|f\| \equiv \Big(\int |f(x)|^s q(x) dx \Big)^{1/s} < \infty \}$$

be a Banach space. The map $W \ni w \mapsto f(x,w)$ is an $L^s(q)$ valued analytic function.
(4) A nonnegative function $K(w)$ is defined as

$$K(w) = \mathbb{E}_X[f(X,w)].$$

The set W_ε is defined as

$$W_\varepsilon = \{w \in W \ ; \ K(w) \leq \varepsilon\}.$$

It is assumed that there exist constants $\varepsilon, c > 0$ such that

$$(\forall w \in W_\varepsilon) \quad \mathbb{E}_X[f(X,w)] \geq c\, \mathbb{E}_X[f(X,w)^2]. \tag{13}$$

Remark 3 *In ordinary learning problems, if the true distribution is regular for or realizable by a learning machine, then assumptions (1), (2), (3) and (4) are satisfied, and the results of the present paper hold. If the true distribution is singular for and unrealizable by a learning machine, then assumption (4) is satisfied in some cases but not in other cases. If the assumption (4) is not satisfied, then the Bayes generalization and training errors may have asymptotic behaviors other than those described in Lemma 1 (Watanabe, 2010d).*

The investigation of cross-validation in singular learning machines requires singular learning theory. In previous papers, we obtained the following lemma.

Lemma 1 *Assume that assumptions (1), (2), (3), and (4) are satisfied. Then, the followings hold.*
(1) Three random variables $nB_g(n)$, $nB_t(n)$, and $V(n)$ converge in law, when n tends to infinity. In addition, the expectation values of these variables converge.
(2) For $k = 1,2,3,4$, we define

$$M_k(n) \equiv \sup_{|\alpha| \leq 1+\beta} \mathbb{E}\Big[\frac{1}{n} \sum_{i=1}^{n} \frac{\mathbb{E}_w[|f(X_i,w)|^k \exp(\alpha f(X_i,w))]}{\mathbb{E}_w[\exp(\alpha f(X_i,w))]} \Big],$$

where $\mathbb{E}[\ ]$ gives the average over all sets of training samples. Then,

$$\mathrm{limsup}_{n\to\infty} \Big(n^{k/2}\, M_k(n) \Big) < \infty. \tag{14}$$

(3) The expectation value of the Bayes generalization loss is asymptotically equal to the widely applicable information criterion,

$$\mathbb{E}[B_g L(n)] = \mathbb{E}[WAIC(n)] + o(\frac{1}{n}). \tag{15}$$

Proof For the case in which $q(x)$ is realizable by and singular for $p(x|w)$, this lemma was proven in Watanabe (2010a) and Watanabe (2009). In fact, the proof of Lemma 1 (1) is given in Theorem 1 of Watanabe (2010a). Also Lemma 1 (2) can be proven in the same manner as Equation (32) in Watanabe (2010a) or Equation (6.59) in Watanabe (2009). The proof of Lemma 1 (3) is given in Theorem 2 and the discussion of Watanabe (2010a). For the case in which $q(x)$ is regular for and unrealizable by $p(x|w)$, this lemma was proven in Watanabe (2010b). For the case in which $q(x)$ is singular for and unrealizable by $p(x|w)$, these results can be generalized under the condition that Equation (13) is satisfied (Watanabe, 2010d). ■

3. Bayes Cross-Validation

In this section, we introduce the cross-validation in Bayes learning.

The expectation value $\mathbb{E}_w^{(i)}[\ \ ]$ using the posterior distribution leaving out X_i is defined as

$$\mathbb{E}_w^{(i)}[\ \ ] = \frac{\int (\ \ ) \prod_{j\neq i}^{n} p(X_j|w)^{\beta}\, \varphi(w) dw}{\int \prod_{j\neq i}^{n} p(X_j|w)^{\beta}\, \varphi(w) dw}, \tag{16}$$

where $\prod_{j\neq i}^{n}$ shows the product for $j = 1,2,3,..,n$, which does not include $j = i$. The predictive distribution leaving out X_i is defined as

$$p^{(i)}(x) = \mathbb{E}_w^{(i)}[p(x|w)].$$

The log loss of $p^{(i)}(x)$ when X_i is used as a testing sample is

$$-\log p^{(i)}(X_i) = -\log \mathbb{E}_w^{(i)}[p(X_i|w)].$$

Thus, the log loss of the Bayes cross-validation is defined as the empirical average of them,

$$C_v L(n) = -\frac{1}{n}\sum_{i=1}^{n} \log \mathbb{E}_w^{(i)}[p(X_i|w)]. \tag{17}$$

The random variable $C_v L(n)$ is referred to as the *cross-validation loss*. Since $X_1, X_2, ..., X_n$ are independent training samples, it immediately follows that

$$\mathbb{E}[C_v L(n)] = \mathbb{E}[B_g L(n-1)].$$

Although the two random variables $C_v L(n)$ and $B_g L(n-1)$ are different,

$$C_v L(n) \neq B_g L(n-1),$$

their expectation values coincide with each other by the definition. Using Equation (15), it follows that

$$\mathbb{E}[C_v L(n)] = \mathbb{E}[\mathrm{WAIC}(n-1)] + o(\frac{1}{n}).$$

Therefore, three expectation values $\mathbb{E}[C_v L(n)]$, $\mathbb{E}[B_g L(n-1)]$, and $\mathbb{E}[\mathrm{WAIC}(n-1)]$ are asymptotically equal to each other. The primary goal of the present paper is to clarify the asymptotic behaviors of three random variables, $C_v L(n)$, $B_g L(n)$, and $\mathrm{WAIC}(n)$, when n is sufficiently large.

Remark 4 *In practical applications, the Bayes generalization loss $B_g L(n)$ indicates the accuracy of Bayes estimation. However, in order to calculate $B_g L(n)$, we need the expectation value over the testing sample taken from the unknown true distribution, hence we cannot directly obtain $B_g L(n)$ in practical applications. On the other hand, both the cross-validation loss $C_v L(n)$ and the widely applicable information criterion $WAIC(n)$ can be calculated using only training samples. Therefore, the cross-validation loss and the widely applicable information criterion can be used for model selection and hyperparameter optimization. This is the reason why comparison of these random variables is an important problem in statistical learning theory.*

4. Main Results

In this section, the main results of the present paper are explained. First, we define functional cumulants and describe their asymptotic properties. Second, we prove that both the cross-validation loss and the widely applicable information criterion can be represented by the functional cumulants. Finally, we prove that the cross-validation loss and the widely applicable information criterion are related to the birational invariants.

4.1 Functional Cumulants

Definition 5 *The generating function $F(\alpha)$ of functional cumulants is defined as*

$$F(\alpha) = \frac{1}{n}\sum_{i=1}^{n} \log \mathbb{E}_w[p(X_i|w)^\alpha].$$

The kth order functional cumulant $Y_k(n)$ $(k = 1,2,3,4)$ is defined as

$$Y_k(n) = \frac{d^k F}{d\alpha^k}(0). \tag{18}$$

Then, by definition,

$$\begin{aligned} F(0) &= 0, \\ F(1) &= -B_t L(n), \\ Y_1(n) &= -G_t L(n), \\ Y_2(n) &= V(n)/n. \end{aligned}$$

For simple notation, we use

$$\ell_k(X_i) = \mathbb{E}_w[(\log p(X_i|w))^k] \quad (k = 1,2,3,4).$$

Lemma 2 *Then, the following equations hold:*

$$Y_1(n) = \frac{1}{n}\sum_{i=1}^{n} \ell_1(X_i), \tag{19}$$

$$Y_2(n) = \frac{1}{n}\sum_{i=1}^{n}\Big\{\ell_2(X_i) - \ell_1(X_i)^2\Big\}, \tag{20}$$

$$Y_3(n) = \frac{1}{n}\sum_{i=1}^{n}\Big\{\ell_3(X_i) - 3\ell_2(X_i)\ell_1(X_i) + 2\ell_1(X_i)^3\Big\}, \tag{21}$$

$$Y_4(n) = \frac{1}{n}\sum_{i=1}^{n}\Big\{\ell_4(X_i) - 4\ell_3(X_i)\ell_1(X_i) - 3\ell_2(X_i)^2 + 12\ell_2(X_i)\ell_1(X_i)^2 - 6\ell_1(X_i)^4\Big\}. \tag{22}$$

Moreover,

$$Y_k(n) = O_p(\frac{1}{n^{k/2}}) \quad (k = 2,3,4).$$

In other words,

$$\mathrm{limsup}_{n\to\infty}\mathbb{E}[n^{k/2}\,|Y_k(n)|] < \infty \quad (k = 2,3,4). \tag{23}$$

Proof First, we prove Equations (19) through (22). Let us define

$$g_i(\alpha) = \mathbb{E}_w[p(X_i|w)^\alpha].$$

Then, $g_i(0) = 1$,

$$g_i^{(k)}(0) \equiv \frac{d^k g_i}{d\alpha^k}(0) = \ell_k(X_i) \quad (k = 1,2,3,4),$$

and

$$F(\alpha) = \frac{1}{n}\sum_{i=1}^{n}\log g_i(\alpha).$$

For arbitrary natural number k,

$$\Big(\frac{g_i(\alpha)^{(k)}}{g_i(\alpha)}\Big)' = \frac{g_i(\alpha)^{(k+1)}}{g_i(\alpha)} - \Big(\frac{g_i(\alpha)^{(k)}}{g_i(\alpha)}\Big)\Big(\frac{g_i(\alpha)'}{g_i(\alpha)}\Big).$$

By applying this relation recursively, Equations (19), (20), (21), and (22) are derived. Let us prove Equation (23). The random variables $Y_k(n)$ $(k = 2,3,4)$ are invariant under the transform,

$$\log p(X_i|w) \mapsto \log p(X_i|w) + c(X_i), \tag{24}$$

for arbitrary $c(X_i)$. In fact, by replacing $p(X_i|w)$ by $p(X_i|w)e^{C(X_i)}$, we define

$$\hat{F}(\alpha) = \frac{1}{n}\sum_{i=1}^{n}\log\mathbb{E}_w[p(X_i|w)^\alpha\, e^{\alpha c(X_i)}].$$

Then, the difference between $F(\alpha)$ and $\hat{F}(\alpha)$ is a linear function of α, which vanishes by higher-order differentiation. In particular, by selecting $c(X_i) = -\log p_0(X_i)$, we can show that $Y_k(n)$ $(k = 2,3,4)$ are invariant by the following replacement,

$$\log p(X_i|w) \mapsto f(X_i, w).$$

In other words, $Y_k(n)$ $(n=2,3,4)$ are invarianrt by the replacement,

$$\ell_k(X_i) \mapsto \mathbb{E}_w[f(X_i,w)^k].$$

Using the Cauchy-Schwarz inequality, for $1 \leq k' \leq k$,

$$\mathbb{E}_w[|f(X_i,w)|^{k'}]^{1/k'} \leq \mathbb{E}_w[|f(X_i,w)|^k]^{1/k}.$$

Therefore, for $k=2,3,4$,

$$\mathbb{E}[|Y_k(n)|] \leq \mathbb{E}\Big[\frac{C_k}{n}\sum_{i=1}^n \mathbb{E}_w[|f(X_i,w)|^k]\Big] \leq C_k M_k(n),$$

where $C_2=2, C_3=6, C_4=26$. Then, using Equation (14), we obtain Equation (23). ■

Remark 6 *Using Equation (24) with* $c(X_i) = -\mathbb{E}_w[\log p(X_i|w)]$ *and the normalized function defined as*

$$\ell_k^*(X_i) = \mathbb{E}_w[(\log p(X_i|w) - c(X_i))^k],$$

it follows that

$$\begin{aligned} Y_2(n) &= \frac{1}{n}\sum_{i=1}^n \ell_2^*(X_i), \\ Y_3(n) &= \frac{1}{n}\sum_{i=1}^n \ell_3^*(X_i), \\ Y_4(n) &= \frac{1}{n}\sum_{i=1}^n \Big\{\ell_4^*(X_i) - 3\ell_2^*(X_i)^2\Big\}. \end{aligned}$$

These formulas may be useful in practical applications.

4.2 Bayes Cross-validation and Widely Applicable Information Criterion

We show the asymptotic equivalence of the cross-validation loss $C_v L(n)$ and the widely applicable information criterion $\mathrm{WAIC}(n)$.

Theorem 1 *For arbitrary* $0 < \beta < \infty$, *the cross-validation loss* $C_v L(n)$ *and the widely applicable information criterion* $\mathrm{WAIC}(n)$ *are given, respectively, as*

$$\begin{aligned} C_v L(n) &= -Y_1(n) + \Big(\frac{2\beta-1}{2}\Big)Y_2(n) \\ &\quad -\Big(\frac{3\beta^2-3\beta+1}{6}\Big)Y_3(n) + O_p(\frac{1}{n^2}), \\ \mathrm{WAIC}(n) &= -Y_1(n) + \Big(\frac{2\beta-1}{2}\Big)Y_2(n) \\ &\quad -\frac{1}{6}Y_3(n) + O_p(\frac{1}{n^2}). \end{aligned}$$

Proof First, we consider $C_v L(n)$. From the definitions of $\mathbb{E}_w[\ ]$ and $\mathbb{E}_w^{(i)}[\ ]$, we have

$$\mathbb{E}_w^{(i)}[(\ \ )] = \frac{\mathbb{E}_w[(\ \ )p(X_i|w)^{-\beta}\ ]}{\mathbb{E}_w[p(X_i|w)^{-\beta}\ ]}. \tag{25}$$

Therefore, by the definition of the cross-validation loss, Equation (17),

$$C_v L(n) = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{\mathbb{E}_w[\ p(X_i|w)^{1-\beta}\ ]}{\mathbb{E}_w[\ p(X_i|w)^{-\beta}\ ]}.$$

Using the generating function of functional cumulants $F(\alpha)$,

$$C_v L(n) = F(-\beta) - F(1-\beta). \tag{26}$$

Then, using Lemma 1 (2) for each $k = 2,3,4$, and $|\alpha| < 1+\beta$,

$$\begin{aligned}
\mathbb{E}[|F^{(k)}(\alpha)|] &\leq \mathbb{E}\Big[\frac{C_k}{n}\sum_{i=1}^{n}\frac{\mathbb{E}_w[|f(X_i,w)|^k \exp(\alpha f(X_i,w))]}{\mathbb{E}_w[\exp(\alpha f(X_i,w))]}\Big] \\
&\leq C_k M_k(n),
\end{aligned}$$

where $C_2 = 2, C_3 = 6, C_4 = 26$. Therefore,

$$|F^{(k)}(\alpha)| = O_p(\frac{1}{n^{k/2}}). \tag{27}$$

By Taylor expansion of $F(\alpha)$ among $\alpha = 0$, there exist β^*, β^{**} ($|\beta^*|, |\beta^{**}| < 1+\beta$) such that

$$\begin{aligned}
F(-\beta) &= F(0) - \beta F'(0) + \frac{\beta^2}{2}F''(0) \\
&\quad -\frac{\beta^3}{6}F^{(3)}(0) + \frac{\beta^4}{24}F^{(4)}(\beta^*), \\
F(1-\beta) &= F(0) + (1-\beta)F'(0) + \frac{(1-\beta)^2}{2}F''(0) \\
&\quad +\frac{(1-\beta)^3}{6}F^{(3)}(0) + \frac{(1-\beta)^4}{24}F^{(4)}(\beta^{**}).
\end{aligned}$$

Using $F(0) = 0$ and Equations (26) and (27), it follows that

$$\begin{aligned}
C_v L(n) &= -F'(0) + \frac{2\beta-1}{2}F''(0) \\
&\quad -\frac{3\beta^2 - 3\beta + 1}{6}F^{(3)}(0) + O_p(\frac{1}{n^2}).
\end{aligned}$$

Thus, we have proven the first half of the theorem. For the latter half, by the definitions of $\mathrm{WAIC}(n)$, Bayes training loss, and the functional variance, we have

$$\begin{aligned}
\mathrm{WAIC}(n) &= B_t L(n) + (\beta/n)V(n), \\
B_t L(n) &= -F(1), \\
V(n) &= nF''(0).
\end{aligned}$$

Therefore,

$$\mathrm{WAIC}(n) = -F(1) + \beta F''(0).$$

By Taylor expansion of $F(1)$, we obtain

$$\mathrm{WAIC}(n) = -F'(0) + \frac{2\beta - 1}{2} F''(0) - \frac{1}{6} F^{(3)}(0) + O_p(\frac{1}{n^2}),$$

which completes the proof. ■

From the above theorem, we obtain the following corollary.

Corollary 1 *For arbitrary $0 < \beta < \infty$, the cross-validation loss $C_v L(n)$ and the widely applicable information criterion* $\mathrm{WAIC}(n)$ *satisfy*

$$C_v L(n) = \mathrm{WAIC}(n) + O_p(\frac{1}{n^{3/2}}).$$

In particular, for $\beta = 1$,

$$C_v L(n) = \mathrm{WAIC}(n) + O_p(\frac{1}{n^2}).$$

More precisely, the difference between the cross-validation loss and the widely applicable information criterion is given by

$$C_v L(n) - \mathrm{WAIC}(n) \cong \Big(\frac{\beta - \beta^2}{2}\Big) Y_3(n).$$

If $\beta = 1$,

$$C_v L(n) - \mathrm{WAIC}(n) \cong \frac{1}{12} Y_4(n).$$

4.3 Generalization Error and Cross-validation Error

In the previous subsection, we have shown that the cross-validation loss is asymptotically equivalent to the widely applicable information criterion. In this section, let us compare the Bayes generalization error $B_g(n)$ given in Equation (11) and the cross-validation error $C_v(n)$, which is defined as

$$C_v(n) = C_v L(n) - L_n. \tag{28}$$

We need mathematical concepts, the real log canonical threshold, and the singular fluctuation.

Definition 7 *The zeta function $\zeta(z)$ $(Re(z) > 0)$ of statistical learning is defined as*

$$\zeta(z) = \int K(w)^z \varphi(w) dw,$$

where

$$K(w) = \mathbb{E}_X[f(X,w)]$$

is a nonnegative analytic function. Here, $\zeta(z)$ can be analytically continued to the unique meromorphic function on the entire complex plane $\mathbb{C}$. All poles of $\zeta(z)$ are real, negative, and rational numbers. The maximum pole is denoted as

$$(-\lambda) = \textit{ maximum pole of } \zeta(z). \tag{29}$$

Then, the positive rational number λ is referred to as the real log canonical threshold. The singular fluctuation is defined as

$$\nu = \nu(\beta) = \lim_{n\to\infty} \frac{\beta}{2}\mathbb{E}[V(n)]. \tag{30}$$

Note that the real log canonical threshold does not depend on β, whereas the singular fluctuation is a function of β.

Both the real log canonical threshold and the singular fluctuation are birational invariants. In other words, they are determined by the algebraic geometrical structure of the statistical model. The following lemma was proven in a previous study (Watanabe, 2010a,b,d).

Lemma 3 *The following convergences hold:*

$$\lim_{n\to\infty} n\mathbb{E}[B_g(n)] = \frac{\lambda-\nu}{\beta} + \nu, \tag{31}$$

$$\lim_{n\to\infty} n\mathbb{E}[B_t(n)] = \frac{\lambda-\nu}{\beta} - \nu, \tag{32}$$

Moreover, convergence in probability

$$n(B_g(n) + B_t(n)) + V(n) \to \frac{2\lambda}{\beta} \tag{33}$$

holds.

Proof For the case in which $q(x)$ is realizable by and singular for $p(x|w)$, Equations (31) and (32) were proven by in Corollary 3 in Watanabe (2010a). The equation (33) was given in Corollary 2 in Watanabe (2010a). For the case in which $q(x)$ is regular for $p(x|w)$, these results were proved in Watanabe (2010b). For the case in which $q(x)$ is singular for and unrealizable by $p(x|w)$ they were generalized in Watanabe (2010d). ■

4.3.1 EXAMPLES

If $q(x)$ is regular for and realizable by $p(x|w)$, then $\lambda = \nu = d/2$, where d is the dimension of the parameter space. If $q(x)$ is regular for and unrealizable by $p(x|w)$, then λ and ν are given by Watanabe (2010b). If $q(x)$ is singular for and realizable by $p(x|w)$, then λ for several models are obtained by resolution of singularities (Aoyagi and Watanabe, 2005; Rusakov and Geiger, 2005; Yamazaki and Watanabe, 2003; Lin, 2010; Zwiernik, 2010). If $q(x)$ is singular for and unrealizable by $p(x|w)$, then λ and ν remain unknown constants.

We have the following theorem.

Theorem 2 *The following equation holds:*

$$\lim_{n\to\infty} n\mathbb{E}[C_v(n)] = \frac{\lambda-\nu}{\beta}+\nu,$$

The sum of the Bayes generalization error and the cross-validation error satisfies

$$B_g(n)+C_v(n) = (\beta-1)\frac{V(n)}{n}+\frac{2\lambda}{\beta n}+o_p(\frac{1}{n}).$$

In particular, if $\beta = 1$,

$$B_g(n)+C_v(n) = \frac{2\lambda}{n}+o_p(\frac{1}{n}).$$

Proof By Equation (31),

$$\mathbb{E}[B_g(n-1)] = \Big(\frac{\lambda-\nu}{\beta}+\nu\Big)\frac{1}{n}+o(\frac{1}{n}).$$

Since $\mathbb{E}[C_v(n)] = \mathbb{E}[B_g(n-1)]$,

$$\begin{aligned}\lim_{n\to\infty} n\mathbb{E}[C_v(n)] &= \lim_{n\to\infty} n\mathbb{E}[B_g(n-1)]\\ &= \frac{\lambda-\nu}{\beta}+\nu.\end{aligned}$$

From Equation (33) and Corollary 1,

$$B_t(n) = C_v(n)-\frac{\beta}{n}V(n)+O_p(\frac{1}{n^{3/2}}),$$

and it follows that

$$(B_g(n)+C_v(n)) = (\beta-1)\frac{V(n)}{n}+\frac{2\lambda}{\beta n}+o_p(\frac{1}{n}),$$

which proves the Theorem. ■

This theorem indicates that both the cross-validation error and the Bayes generalization error are determined by the algebraic geometrical structure of the statistical model, which is extracted as the real log canonical threshold. From this theorem, in the strict Bayes case $\beta = 1$, we have

$$\begin{aligned}\mathbb{E}[B_g(n)] &= \frac{\lambda}{n}+o(\frac{1}{n}),\\ \mathbb{E}[C_v(n)] &= \frac{\lambda}{n}+o(\frac{1}{n}),\end{aligned}$$

and

$$B_g(n)+C_v(n) = \frac{2\lambda}{n}+o_p(\frac{1}{n}). \tag{34}$$

Therefore, the smaller cross-validation error $C_v(n)$ is equivalent to the larger Bayes generalization error $B_g(n)$. Note that a regular statistical model is a special example of singular models, hence both Theorems 1 and 2 also hold in regular statistical models. In Watanabe (2009), it was proven that the random variable $nB_g(n)$ converges to a random variable in law. Thus, $nC_v(n)$ converges to a random variable in law. The asymptotic probability distribution of $nB_g(n)$ can be represented using a Gaussian process, which is defined on the set of true parameters, but is not equal to the χ^2 distribution in general.

Remark 8 *The relation given by Equation (34) indicates that, if $\beta = 1$, the variances of $B_g(n)$ and $C_v(n)$ are equal. If the average value $2\nu = \mathbb{E}[V(n)]$ is known, then $B_t(n) + 2\nu/n$ can be used instead of $C_v(n)$, because both average values are asymptotically equal to the Bayes generalization error. The variance of $B_t(n) + 2\nu/n$ is smaller than that of $C_v(n)$ if and only if the variance of $B_t(n)$ is smaller than that of $B_g(n)$. If a true distribution is regular for and realizable by the statistical model, then the variance of $B_t(n)$ is asymptotically equal to that of $B_g(n)$. However, in other cases, the variance of $B_t(n)$ may be smaller or larger than that of $B_g(n)$.*

5. Discussion

Let us now discuss the results of the present paper.

5.1 From Regular to Singular

First, we summarize the regular and singular learning theories.

In regular statistical models, the generalization loss of the maximum likelihood method is asymptotically equal to that of the Bayes estimation. In both the maximum likelihood and Bayes methods, the cross-validation losses have the same asymptotic behaviors. The leave-one-out cross-validation is asymptotically equivalent to the AIC, in both the maximum likelihood and Bayes methods.

On the other hand, in singular learning machines, the generalization loss of the maximum likelihood method is larger than the Bayes generalization loss. Since the generalization loss of the maximum likelihood method is determined by the maximum value of the Gaussian process, the maximum likelihood method is not appropriate in singular models (Watanabe, 2009). In Bayes estimation, we derived the asymptotic expansion of the generalization loss and proved that the average of the widely applicable information criterion is asymptotically equal to the Bayes generalization loss (Watanabe, 2010a). In the present paper, we clarified that the leave-one-out cross-validation in Bayes estimation is asymptotically equivalent to WAIC.

It was proven (Watanabe, 2001a) that the Bayes marginal likelihood of a singular model is different from BIC of a regular model. In the future, we intend to compare the cross-validation and Bayes marginal likelihood in model selection and hyperparameter optimization in singular statistical models.

5.2 Cross- validation and Importance Sampling

Second, let us investigate the cross-validation and the importance sampling cross-validation from a practical viewpoint.

In Theorem 1, we theoretically proved that the leave-one-out cross-validation is asymptotically equivalent to the widely applicable information criterion. In practical applications, we often approx-

imate the posterior distribution using the Markov Chain Monte Carlo or other numerical methods. If the posterior distribution is precisely realized, then the two theorems of the present paper hold. However, if the posterior distribution was not precisely approximated, then the cross-validation might not be equivalent to the widely applicable information criterion.

In Bayes estimation, there are two different methods by which the leave-one-out cross-validation is numerically approximated. In the former method, CV_1 is obtained by realizing all posterior distributions $\mathbb{E}_w^{(i)}[\ ]$ leaving out X_i for $i = 1, 2, 3, ..., n$, and the empirical average

$$CV_1 = -\frac{1}{n}\sum_{i=1}^{n} \log \mathbb{E}_w^{(i)}[p(X_i|w)]$$

is then calculated. In this method, we must realize n different posterior distributions, which requires heavy computational costs.

In the latter method, the posterior distribution leaving out X_i is estimated using the posterior average $\mathbb{E}_w[\ ]$, in the same manner as Equation (25),

$$\mathbb{E}_w^{(i)}[p(X_i|w)] \cong \frac{\mathbb{E}_w[p(X_i|w)\ p(X_i|w)^{-\beta}\ ]}{\mathbb{E}_w[p(X_i|w)^{-\beta}\ ]}.$$

This method is referred to as the importance sampling leave-one-out cross-validation (Gelfand et al., 1992), in which only one posterior distribution is needed and the leave-one-out cross-validation is approximated by CV_2,

$$CV_2 \cong -\frac{1}{n}\sum_{i=1}^{n} \log \frac{\mathbb{E}_w[p(X_i|w)\ p(X_i|w)^{-\beta}\ ]}{\mathbb{E}_w[p(X_i|w)^{-\beta}\ ]}.$$

If the posterior distribution is completely realized, then CV_1 and CV_2 coincide with each other and are asymptotically equivalent to the widely applicable information criterion. However, if the posterior distribution is not sufficiently approximated, then the values CV_1, CV_2, and $\mathrm{WAIC}(n)$ might be different.

The average values using the posterior distribution may sometimes have infinite variances (Peruggia, 1997) if the set of parameters is not compact. Moreover, in singular learning machines, the set of true parameters is not a single point but rather an analytic set, hence we must restrict the parameter space to be compact for well-defined average values. Therefore, we adopted the assumptions in Section 2.3 that the parameter space is compact and the log likelihood function has the appropriate properties. Under these conditions, the observables studied in the present paper have finite variances.

5.3 Comparison with the Deviance Information Criteria

Third, let us compare the deviance information criterion (DIC) (Spiegelhalter et al., 2002) to the Bayes cross-validation and WAIC, because DIC is sometimes used in Bayesian model evaluation. In order to estimate the Bayesian generalization error, DIC is written by

$$DIC_1 = B_t L(n) + \frac{2}{n}\sum_{i=1}^{n} \Big\{ -E_w[\log p(X_i|w)] + \log p(X_i|E_w[w]) \Big\},$$

where the second term of the right-hand side corresponds to the "effective number of parameters" of DIC divided by the number of parameters. Under the condition that the log likelihood ratio function

in the posterior distribution is subject to the χ^2 distribution, a modified DIC was proposed (Gelman et al., 2004) as

$$DIC_2 = B_t L(n) + \frac{2}{n}\Big[E_w[\Big\{\sum_{i=1}^{n}\log p(X_i|w)\Big\}^2] - E_w[\sum_{i=1}^{n}\log p(X_i|w)]^2\Big],$$

the variance of which was investigated previously (Raftery, 2007). Note that DIC_2 is different from WAIC. In a singular learning machine, since the set of optimal parameters is an analytic set, the correlation between different true parameters does not vanish, even asymptotically.

We first derive the theoretical properties of DIC. If the true distribution is regular for the statistical model, then the set of the optimal parameter is a single point w_0. Thus, the difference of $E_w[w]$ and the maximum *a posteriori* estimator is asymptotically smaller than $1/\sqrt{n}$. Therefore, based on the results in Watanabe (2010b), if $\beta = 1$,

$$\mathbb{E}[DIC_1] = L_0 + (3\lambda - 2\nu(1))\frac{1}{n} + o(\frac{1}{n}).$$

If the true distribution is realizable by or regular for the statistical model and if $\beta = 1$, then the asymptotic behavior of DIC_2 is given by

$$\mathbb{E}[DIC_2] = L_0 + (3\lambda - 2\nu(1) + 2\nu'(1))\frac{1}{n} + o(\frac{1}{n}), \tag{35}$$

where $\nu'(1) = (d\nu/d\beta)(1)$. Equation (35) is derived from the relations (Watanabe, 2009, 2010a,b,d),

$$\begin{aligned} DIC_2 &= B_t L(n) - 2\frac{\partial}{\partial\beta}G_t L(n), \\ \mathbb{E}[G_t L(n)] &= L_0 + \Big(\frac{\lambda}{\beta} - \nu(\beta)\Big)\frac{1}{n} + o(\frac{1}{n}), \end{aligned}$$

where $G_t L(n)$ is given by Equation (7).

Next, let us consider the DIC for each case. If the true distribution is regular for and realizable by the statistical model and if $\beta = 1$, then $\lambda = \nu = d/2, \nu'(1) = 0$, where d is the number of parameters. Thus, their averages are asymptotically equal to the Bayes generalization error,

$$\begin{aligned} \mathbb{E}[DIC_1] &= L_0 + \frac{d}{2n} + o(\frac{1}{n}), \\ \mathbb{E}[DIC_2] &= L_0 + \frac{d}{2n} + o(\frac{1}{n}). \end{aligned}$$

In this case, the averages of DIC_1, DIC_2, CV_1, CV_2, and WAIC have the same asymptotic behavior.

If the true distribution is regular for and unrealizable by the statistical model and if $\beta = 1$, then $\lambda = d/2$, $\nu = (1/2)\mathrm{tr}(IJ^{-1})$, and $\nu'(1) = 0$ (Watanabe, 2010b), where I is the Fisher information matrix at w_0, and J is the Hessian matrix of $L(w)$ at $w = w_0$. Thus, we have

$$\begin{aligned} \mathbb{E}[DIC_1] &= L_0 + \Big(\frac{3d}{2} - \mathrm{tr}(IJ^{-1})\Big)\frac{1}{n} + o(\frac{1}{n}), \\ \mathbb{E}[DIC_2] &= L_0 + \Big(\frac{3d}{2} - \mathrm{tr}(IJ^{-1})\Big)\frac{1}{n} + o(\frac{1}{n}). \end{aligned}$$

In this case, as shown in Lemma 3, the Bayes generalization error is given by $L_0 + d/(2n)$ asymptotically, and so the averages of the deviance information criteria are not equal to the average of the Bayes generalization error.

If the true distribution is singular for and realizable by the statistical model and if $\beta = 1$, then

$$\begin{aligned} \mathbb{E}[DIC_1] &= C + o(1), \\ \mathbb{E}[DIC_2] &= L_0 + (3\lambda - 2\nu(1) + 2\nu'(1))\frac{1}{n} + o(\frac{1}{n}), \end{aligned} \tag{36}$$

where C ($C \neq L_0$) is, in general, a constant. Equation (36) is obtained because the set of true parameters in a singular model is not a single point, but rather an analytic set, so that, in general, the average $E_w[w]$ is not contained in the neighborhood of the set of the true parameters. Hence the averages of the deviance information criteria are not equal to those of the Bayes generalization error.

The averages of the cross-validation loss and WAIC have the same asymptotic behavior as that of the Bayes generalization error, even if the true distribution is unrealizable by or singular for the statistical model. Therefore, the deviance information criteria are different from the cross-validation and WAIC, if the true distribution is singular for or unrealizable by the statistical model.

5.4 Experiment

In this section, we describe an experiment. The purpose of the present paper is to clarify the theoretical properties of the cross-validation and the widely applicable information criterion. An experiment was conducted in order to illustrate the main theorems.

Let $x, y \in \mathbb{R}^3$. We considered a statistical model defined as

$$p(x,y|w) = \frac{s(x)}{(2\pi\sigma^2)^{3/2}} \exp(-\frac{\|y - R_H(x,w)\|^2}{2\sigma^2}),$$

where $\sigma = 0.1$ and $s(x)$ is $\mathcal{N}(0, 2^2 I)$. Here, $\mathcal{N}(m,A)$ exhibits a normal distribution with the average vector m and the covariance matrix A, and I is the identity matrix. Note that the distribution $s(x)$ was not estimated. We used a three-layered neural network,

$$R_H(x,w) = \sum_{h=1}^{H} a_h \tanh(b_h \cdot x),$$

where the parameter was

$$w = \{(a_h \in \mathbb{R}^3, b_h \in \mathbb{R}^3)\ ;\ h = 1,2,...,H\} \in \mathbb{R}^{6H}.$$

In the experiment, a learning machine with $H = 3$ was used and the true distribution was set with $H = 1$. The parameter that gives the distribution is denoted as w_0, which denotes the parameters of both models $H = 1, 3$. Then, $R_H(x,w_0) = R_{H_0}(x,w_0)$. Under this condition, the set of true parameters

$$\{w \in W; p(x|w) = p(x|w_0)\}$$

is not a single point but an analytic set with singularities, resulting that the regularity condition is not satisfied. In this case, the log density ratio function is equivalent to

$$f(x,y,w) = \frac{1}{2\sigma^2}\Big\{\|y - R_H(x,w)\|^2 - \|y - R_H(x,w_0)\|^2\Big\}.$$

In this model, although the Bayes generalization error is not equal to the average square error

$$\mathrm{SE}(n) = \frac{1}{2\sigma^2}\mathbb{E}\mathbb{E}_X\Big[\parallel R_H(X,w_0) - \mathbb{E}_w[R_H(X,w)] \parallel^2\Big],$$

asymptotically $\mathrm{SE}(n)$ and $B_g(n)$ are equal to each other (Watanabe, 2009).

The prior distribution $\varphi(w)$ was set as $\mathcal{N}(0, 10^2 I)$. Although this prior does not have compact support mathematically, it can be understood in the experiment that the support of $\varphi(w)$ is essentially contained in a sufficiently large compact set.

In the experiment, the number of training samples was fixed as $n = 200$. One hundred sets of 200 training samples each were obtained independently. For each training set, the strict Bayes posterior distribution $\beta = 1$ was approximated by the Markov chain Monte Carlo (MCMC) method. The Metropolis method, in which each random trial was taken from $\mathcal{N}(0, (0.005)^2 I)$, was applied, and the average exchanging ratio was obtained as approximately 0.35. After 100,000 iterations of Metropolis random sampling, 200 parameters were obtained in every 100 sampling steps. For a fixed training set, by changing the initial values and the random seeds of the software, the same MCMC sampling procedures were performed 10 times independently, which was done for the purpose of minimizing the effect of the local minima. Finally, for each training set, we obtained $200 \times 10 = 2,000$ parameters, which were used to approximate the posterior distribution.

Table 2 shows the experimental results. We observed the Bayes generalization error $BG = B_g(n)$, the Bayes training error $BT = B_t(n)$, importance sampling leave-one-out cross-validation $CV = CV_2 - L_n$, the widely applicable information criterion $\mathrm{WAIC} = \mathrm{WAIC}(n) - L_n$, two deviance information criteria, namely, $DIC1 = DIC_1 - L_n$ and $DIC2 = DIC_2 - L_n$, and the sum $BG + CV = B_g(n) + C_v(n)$. The values AVR and STD in Table 2 show the average and standard deviation of one hundred sets of training data, respectively. The original cross-validation CV_1 was not observed because the associated computational cost was too high.

The experimental results reveal that the average and standard deviation of BG were approximately the same as those of CV and WAIC, which indicates that Theorem 1 holds. The real log canonical threshold, the singular fluctuation, and its derivative of this case were estimated as

$$\begin{aligned} \lambda &\approx 5.6, \\ \nu(1) &\approx 7.9, \\ \nu'(1) &\approx 3.6. \end{aligned}$$

Note that, if the true distribution is regular for and realizable by the statistical model, $\lambda = \nu(1) = d/2 = 9$ and $\nu'(1) = 0$. The averages of the two deviance information criteria were not equal to that of the Bayes generalization error. The standard deviation of $BG + CV$ was smaller than the standard deviations of BG and CV, which is in agreement with Theorem 2.

Note that the standard deviation of BT was larger than those of CV and WAIC, which indicates that, even if the average value $\mathbb{E}[C_v(n) - B_t(n)] = 2\nu/n$ is known and an alternative cross-validation, such as the AIC,

$$CV_3 = B_t L(n) + 2\nu/n,$$

is used, then the variance of $CV_3 - L_n$ was larger than the variances of $C_v L(n) - L_n$ and $\mathrm{WAIC}(n) - L_n$.

	BG	*BT*	*CV*	WAIC	*DIC*1	*DIC*2	*BG*+*CV*
AVR	0.0264	-0.0511	0.0298	0.0278	-35.1077	0.0415	0.0562
STD	0.0120	0.0165	0.0137	0.0134	19.1350	0.0235	0.0071

Table 2: Average and standard deviation

	BG	*BT*	*CV*	WAIC	*DIC*1	*DIC*2	*BG*+*CV*
BG	1.000	-0.854	-0.854	-0.873	0.031	-0.327	0.043
BT		1.000	0.717	0.736	0.066	0.203	-0.060
CV			1.000	0.996	-0.087	0.340	0.481
WA				1.000	-0.085	0.341	0.443
*DIC*1					1.000	-0.069	-0.115
*DIC*2						1.000	0.102

Table 3: Correlation matrix

Table 3 shows the correlation matrix for several values. The correlation between *CV* and WAIC was 0.996, which indicates that Theorem 1 holds. The correlation between *BG* and *CV* was -0.854, and that between *BG* and WAIC was -0.873, which corresponds to Theorem 2.

The accuracy of numerical approximation of the posterior distribution depends on the statistical model, the true distribution, the prior distribution, the Markov chain Monte Carlo method, and the experimental fluctuation. In the future, we intend to develop a method by which to design experiments. The theorems proven in the present paper may be useful in such research.

5.5 Birational Invariant

Finally, we investigate the statistical problem from an algebraic geometrical viewpoint.

In Bayes estimation, we can introduce an analytic function of the parameter space $g : U \to W$,

$$w = g(u).$$

Let $|g'(u)|$ be its Jacobian determinant. Note that the inverse function g^{-1} is not needed if g satisfies the condition that $\{u \in U; |g'(u)| = 0\}$ is a measure zero set in U. Such a function g is referred to as a birational transform. It is important that, by the transform,

$$\begin{aligned} p(x|w) &\mapsto p(x|g(u)), \\ \varphi(w) &\mapsto \varphi(g(u))|g'(u)|, \end{aligned}$$

the Bayes estimation on W is equivalent to that on U. A constant defined for a set of statistical models and a prior is said to be a birational invariant if it is invariant under such a transform $w = g(u)$.

The real log canonical threshold λ is a birational invariant (Atiyah, 1970; Hiroanaka, 1964; Kashiwara, 1976; Kollór et al., 1998; Mustata, 2002; Watanabe, 2009) that represents the algebraic geometrical relation between the set of parameters W and the set of the optimal parameters W_0. Although the singular fluctuation is also a birational invariant, its properties remain unknown. In the present paper, we proved in Theorem 1 that

$$\mathbb{E}[B_g L(n)] = \mathbb{E}[C_v L(n)] + o(1/n). \tag{37}$$

On the other hand, in Theorem 2, we proved that

$$B_g(n) + C_v(n) = \frac{2\lambda}{n} + o_p(1/n). \tag{38}$$

In model selection or hyperparameter optimization, Equation (37) shows that minimization of the cross-validation makes the generalization loss smaller on average. However, Equation (38) shows that minimization of the cross-validation does not ensure minimum generalization loss. The widely applicable information criterion has the same property as the cross-validation. The constant λ appears to exhibit a bound, which can be attained by statistical estimation for a given pair of a statistical model and a prior distribution. Hence, clarification of the algebraic geometrical structure in statistical estimation is an important problem in statistical learning theory.

6. Conclusion

In the present paper, we have shown theoretically that the leave-one-out cross-validation in Bayes estimation is asymptotically equal to the widely applicable information criterion and that the sum of the cross-validation error and the generalization error is equal to twice the real log canonical threshold divided by the number of training samples. In addition, we clarified that cross-validation and the widely applicable information criterion are different from the deviance information criteria. This result indicates that, even in singular statistical models, the cross-validation is asymptotically equivalent to the information criterion, and that the asymptotic properties of these models are determined by the algebraic geometrical structure of a statistical model.

References

H. Akaike. A new look at the statistical model identification. *IEEE Trans. on Automatic Control*, 19:716-723, 1974.

S. Amari. A universal theorem on learning curves, *Neural Networks*, 6(2):161-166, 1993.

M. Aoyagi. Stochastic complexity and generalization error of a restricted Boltzmann machine in Bayesian estimation. *Journal of Machine Learning Research*, 11:1243-1272, 2010.

M. Aoyagi, S. Watanabe. Stochastic complexities of reduced rank regression in Bayesian estimation. *Neural Networks*, 18(7):924-933, 2005.

M.F. Atiyah. Resolution of singularities and division of distributions. *Communications of Pure and Applied Mathematics*, 13:145-150. 1970.

M.W. Browne. Cross-Validation Methods. *Journal of Mathematical Psychology*, 44:108-132, 2000.

H. Cramer. *Mathematical Methods of Statistics*. Princeton University Press, 1949.

M. Drton, B. Sturmfels, and S. Sullivant. *Lecures on Algebraic Statistics*. Birkhäuser, Berlin, 2009.

S. Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70:320-328, 1975.

I.M. Gelfand and G.E. Shilov. *Generalized Functions*. Academic Press, San Diego, 1964.

A.E. Gelfand, D.K. Dey, H. Chang. Model determination using predictive distributions with implementation via sampling-based method. *Bayesian Statistics*, 4:147-167, Oxford University Press, Oxford, 1992.

A. Gelman, J.B. Carlin, H.S. Stern, D.B. Rubin. *Bayesian Data Analysis*. Chapman & Hall CRC, Boca Raton, 2004.

K. Hagiwara. On the problem in model selection of neural network regression in overrealizable scenario. *Neural Computation*, 14:1979-2002, 2002.

J. A. Hartigan. A failure of likelihood asymptotics for normal mixtures. In Proceedings of the *Barkeley Conference in Honor of J. Neyman and J. Kiefer,* Vol. 2, pages 807–810, 1985.

H. Hironaka. Resolution of singularities of an algebraic variety over a field of characteristic zero. *Annals of Mathematics,* 79:109-326, 1964.

M. Kashiwara. B-functions and holonomic systems. *Inventiones Mathematicae,* 38:33-53, 1976.

J. Kollór, S.Mori, C.H.Clemens, A.Corti. *Birational Geometry of Algebraic Varieties*. Cambridge Tract in Mathematics Cambridge University Press, Cambridge, 1998.

C.I. Mosier. Problems and designs of cross-validation. *Educational and Psychological Measurement*, 11:5-11, 1951.

M. Mustata. Singularities of pairs via jet schemes. *Journal of the American Mathematical Society*, 15:599-615. 2002.

E. Levin, N. Tishby, S.A. Solla. A statistical approaches to learning and generalization in layered neural networks. *Proceedings of IEEE,* 78(10):1568-1574. 1990.

S. Lin. Asymptotic approximation of marginal likelihood integrals. *arXiv:1003.5338*, 2010.

H. Linhart, W. Zucchini. *Model Selection*. John Wiley and Sons, New York, 1986.

K. Nagata and S. Watanabe, Asymptotic behavior of exchange ratio in exchange Monte Carlo method. *International Journal of Neural Networks*, 21(7):980-988, 2008.

T. Oaku. Algorithms for the b-function and D-modules associated with a polynomial. *Journal of Pure Applied Algebra,* 117:495-518, 1997.

M. Peruggia. On the variability of case-detection importance sampling weights in the Bayesian linear model. *Journal of American Statistical Association*, 92:199-207, 1997.

A.E. Raftery, M.A. Newton, J.M. Satagopan, P.N. Krivitsly. Estimating the integrated likelihood via posterior simulation using the harmonic mean identity. *Bayesian Statistics*, 8:1-45, Oxford University Press, Oxford, 2007.

D. Rusakov, D. Geiger. Asymptotic model selection for naive Bayesian network. *Journal of Machine Learning Research*. 6:1-35, 2005.

M. Saito. On real log canonical thresholds, *arXiv:0707.2308v1,* 2007.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics,* 6(2):461-464. 1978.

D.J. Spiegelhalter, N.G. Best, B.P. Carlin, A. Linde. Bayesian measures of model complexity and fit. *Journal of Royal Statistical Society, Series B*, 64(4):583-639, 2002.

M. Stone. An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society*. 39(B):44-47, 1977.

A. Takemura, T.Kuriki. On the equivalence of the tube and Euler characteristic methods for the distribution of the maximum of the gaussian fields over piecewise smooth domains. *Annals of Applied Probability,* 12(2):768-796, 2002.

A. W. van der Vaart, J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer, 1996.

A. Vehtari, J. Lampinen. Bayesian Model Assessment and Comparison Using Cross-Validation Predictive Densities. *Neural Computation*, 14(10):2439-2468, 2002.

S. Watanabe. Generalized Bayesian framework for neural networks with singular Fisher information matrices. In the *Proceedings of the International Symposium on Nonlinear Theory and Its applications,* pages 207–210, 1995.

S. Watanabe. Algebraic analysis for nonidentifiable learning machines. *Neural Computation*, 13(4):899-933, 2001a.

S. Watanabe. Algebraic geometrical methods for hierarchical learning machines. *Neural Networks*. 14(8):1049-1060, 2001b.

S. Watanabe. Almost all learning machines are singular. In the *Proceedings of the IEEE Int. Conf. FOCI*, pages 383–388, 2007.

S. Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, Cambridge, UK, 2009.

S. Watanabe. Equations of states in singular statistical estimation. *Neural Networks*. 23(1):20-34, 2010a.

S. Watanabe. Equations of states in statistical learning for an unrealizable and regular case. *IEICE Transactions*. E93A(3):617-626, 2010b.

S. Watanabe. A limit theorem in singular regression problem. *Advanced Studies of Pure Mathematics*. 57:473-492, 2010c.

S. Watanabe. Asymptotic learning curve and renormalizable condition in statistical learning theory. *Journal of Physics Conference Series*, 233, No.012014, 2010d.

K. Yamazaki, S. Watanabe. Singularities in mixture models and upper bounds of stochastic complexity. *Neural Networks*. 16(7):1029-1038, 2003.

K. Yamazaki, S. Watanabe. Algebraic geometry and stochastic complexity of hidden Markov models. *Neurocomputing*, 69:62-84, 2005.

P. Zwiernik. An asymptotic approximation of the marginal likelihood for general Markov models. *arXiv:1012.0753v1*, 2010.

Journal of Machine Learning Research 11 (2010) 3595-3646 Submitted 3/10; Revised 9/10; Published 12/10

PAC-Bayesian Analysis of Co-clustering and Beyond

Yevgeny Seldin* SELDIN@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstraße 38
72076 Tübingen, Germany

Naftali Tishby TISHBY@CS.HUJI.AC.IL
School of Computer Science and Engineering
Interdisciplinary Center for Neural Computation
The Hebrew University of Jerusalem, Israel

Editor: Inderjit S. Dhillon

Abstract

We derive PAC-Bayesian generalization bounds for supervised and unsupervised learning models based on clustering, such as co-clustering, matrix tri-factorization, graphical models, graph clustering, and pairwise clustering.[1] We begin with the analysis of co-clustering, which is a widely used approach to the analysis of data matrices. We distinguish among two tasks in matrix data analysis: discriminative prediction of the missing entries in data matrices and estimation of the joint probability distribution of row and column variables in co-occurrence matrices. We derive PAC-Bayesian generalization bounds for the expected out-of-sample performance of co-clustering-based solutions for these two tasks. The analysis yields regularization terms that were absent in the previous formulations of co-clustering. The bounds suggest that the expected performance of co-clustering is governed by a trade-off between its empirical performance and the mutual information preserved by the cluster variables on row and column IDs. We derive an iterative projection algorithm for finding a local optimum of this trade-off for discriminative prediction tasks. This algorithm achieved state-of-the-art performance in the MovieLens collaborative filtering task. Our co-clustering model can also be seen as matrix tri-factorization and the results provide generalization bounds, regularization terms, and new algorithms for this form of matrix factorization.

The analysis of co-clustering is extended to tree-shaped graphical models, which can be used to analyze high dimensional tensors. According to the bounds, the generalization abilities of tree-shaped graphical models depend on a trade-off between their empirical data fit and the mutual information that is propagated up the tree levels.

We also formulate weighted graph clustering as a prediction problem: given a subset of edge weights we analyze the ability of graph clustering to predict the remaining edge weights. The analysis of co-clustering easily extends to this problem and suggests that graph clustering should optimize the trade-off between empirical data fit and the mutual information that clusters preserve on graph nodes.

Keywords: matrix tri-factorization, graphical models, graph clustering, pairwise clustering, combinatorial priors, density estimation

---

∗. Also in the School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel.

1. This paper is based on Seldin and Tishby (2008, 2009) and Seldin (2009, 2010).

1. Introduction

Structure learning and, in particular, clustering is an important and long-standing problem in science. In many situations it has to be performed based on a limited data sample and with no or limited supervision. A natural question that arises in this context is to what extent the inferred structure is a reflection of a "true" structure underlying the data or a mere artifact of the learning model and/or statistical fluctuation of the finite sample. But is there a "true structure" in the first place? Consider the following example: assume we have a bag of blocks which we can cluster by multiple parameters, such as shape, color, material they are made of, and so on. All these possibilities are equally plausible and asking whether a clustering of blocks by shape is better or worse than a clustering of blocks by color makes no sense. But the absence of an objective comparison criterion for outputs of two structure learning algorithms poses a serious obstacle for their evaluation and the advancement of unsupervised learning in general.

We argue that one does not learn structure for its own sake, but rather to facilitate solving some higher level task. By evaluating the contribution of structure learning to the solution of the higher level task it is possible to derive an objective comparison of the utility of different structures in the context of this specific task. Returning to the bag of blocks example, if we know that after clustering the blocks we will have to pack them into a box, then the clustering of blocks by shape is much more useful than the clustering of blocks by color, since packing is indifferent to color. We can further measure the amount of time that different clusterings saved us in the packing task and thereby obtain an objective numerical evaluation of the utility of clustering of blocks by different parameters in this context. Moreover, by repeating the experiment several times (or by some more intelligent analysis) it is possible to provide generalization guarantees on how much time this or other clustering algorithm is expected to save us in the packing task in the future.

Since in any non-trivial data many structures coexist simultaneously, "blind" unsupervised learning without specification of its potential application is doomed to failure in the general case. This is because the potential application (or range of applications) can make any property or element of the structure either decisive or completely irrelevant for the task, and hence render it useful or useless for identification by unsupervised learning. The need to consider unsupervised learning within the context of its subsequent application has been pointed out by many researchers, especially those concerned with practical applications of these methods (Guyon et al., 2009). In the present paper we reformulate traditional unsupervised learning problems as prediction problems and then adapt well-developed tools from supervised learning to provide generalization bounds on their expected out-of-sample performance. We start with the problem of co-clustering, but then show that our approach to problem formulation is applicable to and can be analyzed in a much broader range of applications.

Co-clustering is a widely used method for analysis of data in matrix form by simultaneous clustering of rows and columns of the matrix (Banerjee et al., 2007). Here we focus solely on co-clustering solutions that result in a grid form partition of the data matrix. This form of co-clustering is also known as partitional co-clustering (Banerjee et al., 2007), checkerboard bi-clustering (Cheng and Church, 2000; Kluger et al., 2003), grid clustering (Devroye et al., 1996; Seldin and Tishby, 2008, 2009), and box clustering. Note that some authors use the terms co-clustering and bi-clustering to refer to a simultaneous grouping of rows and columns that does not result in a grid-form partition of the whole data matrix (Hartigan, 1972; Madeira and Oliveira, 2004), but these forms of partitions are not discussed in this work. Note as well that this paper considers soft as-

signments of rows and columns to their clusters, while using two-level generative models such as those discussed in Dhillon et al. (2003) and Banerjee et al. (2007) for hard assignments. Recently, Bayesian approaches to co-clustering have been suggested, for example see Shan and Banerjee (2008), Salakhutdinov and Mnih (2008), Shafiei and Milios (2006), Wang et al. (2009) and Lashkari and Golland (2009), which consider mixed memberships by introducing an additional level to the generative process. However, three-level generative models require approximate inference methods such as variational inference or Markov Chain Monte Carlo, whereas the two-level model for discriminative prediction discussed here can be learned by iterative projections. The analysis presented here is not limited to two-dimensional data matrices, but holds for higher dimensional tensors as well. A three-level Bayesian approach to clustered tensor factorization was recently presented by Sutskever et al. (2009).

In the past decade co-clustering has successfully been applied in multiple domains, including clustering of documents and words in text mining (Slonim and Tishby, 2000; El-Yaniv and Souroujon, 2001; Dhillon et al., 2003; Takamura and Matsumoto, 2003), genes and experimental conditions in bioinformatics (Cheng and Church, 2000; Cho et al., 2004; Kluger et al., 2003; Cho and Dhillon, 2008), tokens and contexts in natural language processing (Freitag, 2004; Rohwer and Freitag, 2004; Li and Abe, 1998), viewers and movies in recommender systems (George and Merugu, 2005; Seldin et al., 2007; Salakhutdinov and Mnih, 2008; Seldin, 2009), etc. In Seldin et al. (2007) and Seldin and Tishby (2009) it was pointed out that there are actually two different classes of problems that are solved with co-clustering that correspond to two different high-level tasks and should be analyzed separately. The first class of problems are discriminative prediction tasks, one typical representative of which is collaborative filtering (Herlocker et al., 2004). In collaborative filtering, the analyst is given a matrix of viewers by movies with ratings, for example, on a five-star scale, attributed by the viewers to the movies. The matrix is usually sparse, as most viewers have not seen all the movies. In this problem the task is usually to predict the missing entries. We assume that there is some unknown probability distribution $p(x_1, x_2, y)$ over the triplets of viewer x_1, movie x_2, and rating y. The goal is to build a discriminative predictor $q(y|x_1, x_2)$ that given a pair of viewer x_1 and movie x_2 will predict the expected rating y. A natural form of evaluation of such predictors, no matter whether they are based on co-clustering or not, is to evaluate the expected loss $\mathbb{E}_{p(x_1,x_2,y)}\mathbb{E}_{q(y'|x_1,x_2)}l(Y,Y')$, where $l(y,y')$ is an externally provided loss function for predicting y' instead of y. In Section 3 we provide this analysis for co-clustering-based predictors. The analysis can be used not only to construct co-clustering solutions to this problem, but also to conduct a theoretical comparison of the co-clustering-based approach to this problem with other possible approaches.

The second class of problems, which are solved using co-clustering, are problems of estimation of a joint probability distribution in co-occurrence data analysis. A typical example of this kind of problem is the analysis of word-document co-occurrence matrices in text mining (Slonim and Tishby, 2000; El-Yaniv and Souroujon, 2001; Dhillon et al., 2003). Word-document co-occurrence matrices are matrices of words by documents where the number of times each word occurred in each document is counted in the corresponding entries. If normalized, such a matrix can be regarded as an empirical joint probability distribution of words and documents. To illustrate the difference between co-occurrence data and the data considered in discriminative prediction tasks, we point out that the ratings in the collaborative filtering example are functions of viewer and movie ID pairs and they do not depend on other viewers or movies. By contrast, in co-occurrence data the joint probability (or the number of co-occurrence events) is normalized by the size of the corpus and thus depends on

the whole subset of words and documents considered (or the size of the corpus if no normalization is applied).

Although many researchers have analyzed co-occurrence data by clustering similar words and similar documents (Slonim and Tishby, 2000; El-Yaniv and Souroujon, 2001; Dhillon et al., 2003; Takamura and Matsumoto, 2003), or by using topic models (Steyvers and Griffiths, 2006; Blei and Lafferty, 2009) and other approaches, no clear learning task in this problem has been defined and it remains difficult to compare different approaches or perform model order selection. In Seldin and Tishby (2009) one possible way of defining a high-level task for this problem was suggested. It was assumed that the observed co-occurrence matrix was drawn from an unknown joint probability distribution $p(x_1, x_2)$ of words x_1 and documents x_2. The suggested task was an estimation of this joint probability distribution based on the observed sample. In such a formulation, the quality of an estimator $q(x_1, x_2)$ for $p(x_1, x_2)$ can be measured by $-\mathbb{E}_{p(x_1,x_2)} \ln q(X_1, X_2)$, where the choice of the logarithmic loss is natural in the context of density estimation. In particular, it corresponds to the expected code length of an encoder that uses $q(x_1, x_2)$ to encode samples generated by $p(x_1, x_2)$ (Cover and Thomas, 1991). In Section 3 we provide an analysis of this quantity for co-clustering-based density estimators. Similar to the case with co-clustering-based discriminative predictors, the analysis serves to perform model order selection in this problem. It further enables a theoretical comparison of the co-clustering-based approach to this problem with other possible approaches.

For the purpose of analysis and derivation of generalization bounds for the above two problems we found it convenient to apply the PAC-Bayesian framework (McAllester, 1998, 1999), which is reviewed in Section 2. Similar to the Probably Approximately Correct (PAC) learning model (Valiant, 1984), PAC-Bayesian bounds pose no assumptions or restrictions on the distribution that generates the data (apart from the usual assumption that the data are independent and identically distributed (i.i.d.) and that the train and test distributions are the same). However, unlike the usual PAC bounds, where the whole hypothesis space is characterized by its Vapnik-Chervonenkis (VC) dimension (Vapnik and Chervonenkis, 1968, 1971), PAC-Bayesian bounds apply a non-uniform treatment of the hypotheses by introducing a prior distribution over the hypothesis space. For example, within the class of decision trees a preference for shallow trees can be given by assigning a higher prior. If a good prior over the hypothesis space can be designed, the tightness of the bounds can be improved considerably. As shown in the literature, PAC-Bayesian analysis is able to provide practically useful bounds that in some cases are only 10%-20% away from the test error (Langford, 2005; Seldin and Tishby, 2008; Seldin, 2009; Germain et al., 2009).

Originally, PAC-Bayesian bounds were derived for classification tasks. They have been applied in the analysis of decision trees (Mansour and McAllester, 2000), Support Vector Machines (SVMs) (Langford and Shawe-Taylor, 2002; McAllester, 2003; Langford, 2005; Ambroladze et al., 2007; Crammer et al., 2009; Germain et al., 2009), transductive learning (Derbeko et al., 2004), structured prediction (Bartlett et al., 2005; McAllester, 2007), and other supervised learning models. In Seldin and Tishby (2009) we introduced PAC-Bayesian analysis to discrete density estimation. Recently, Higgs and Shawe-Taylor (2010) applied PAC-Bayesian analysis to continuous density estimation. In Section 3 we present the PAC-Bayesian analysis of discriminative prediction and density estimation with co-clustering. According to the derived bounds, the generalization performance of co-clustering-based models depends on a trade-off between their empirical performance and the mutual information that the clusters preserve on the observed parameters (row and column IDs). The mutual information term introduces model regularization that was absent in the previous formulations of co-clustering (Dhillon et al., 2003; Banerjee et al., 2007). We further suggest

algorithms for optimization of the trade-off in Section 4. In Section 5, we achieve state-of-the-art performance in the prediction of missing ratings in the MovieLens collaborative filtering data set by optimization of the trade-off.

The co-clustering models analyzed here are tightly related to matrix tri-factorization—a 3-factor decomposition of a matrix of the form $A \approx Q_1^T F Q_2$ (Banerjee et al., 2007; Ding et al., 2006; Yoo and Choi, 2009a,b) and to Tucker decomposition of higher dimensional tensors (Kim and Choi, 2007). This relation is discussed in Section 6. We point out that similar to co-clustering itself there are at least two different forms of matrix tri-factorization; one that corresponds to discriminative prediction tasks, such as collaborative filtering, and the other which is more appropriate for co-occurrence data analysis. In the first case A can be arbitrary, whereas in the second case the input matrix A is a joint probability distribution matrix (its entries are non-negative and sum up to one). At the technical level, in discriminative prediction tasks Q_1 and Q_2 are right stochastic matrices (their rows sum up to one) and F is arbitrary, whereas in co-occurrence data analysis Q_1 and Q_2 are left stochastic matrices (their columns sum up to one) and F is a joint probability distribution matrix (in the cluster product space). Our analysis provides generalization bounds, regularization terms, and new algorithms for both forms of matrix tri-factorization.

Co-clustering can also be regarded as a simple graphical model. In Section 7 we suggest how to extend our analysis to more general tree-shaped graphical models. Such graphical models can be useful to treat the curse of dimensionality in the analysis of high dimensional tensors. This also provides a new perspective on learning graphical models: instead of learning a graphical model that fits the training data, the approach suggests optimizing the model's ability to predict new observations. In Sections 3 and 7 it is demonstrated that PAC-Bayesian bounds are able to take advantage of the factor form of graphical models and provide bounds that depend on the sizes of the cliques of graphical models and the amount of mutual information that is propagated up the tree levels.

In Section 8 we extend our approach to the formulation of unsupervised learning problems as prediction problems to graph clustering and pairwise clustering (the latter is equivalent to clustering of a weighted graph, where edge weights correspond to pairwise distances). We formulate weighted graph clustering as a prediction problem: given a sample of edge weights we analyze the ability of graph clustering to predict the remaining edge weights. We adapt the PAC-Bayesian analysis of co-clustering to derive a PAC-Bayesian generalization bound for graph clustering. The bound shows that graph clustering should optimize a trade-off between empirical data fit and the mutual information that clusters preserve on the graph nodes. A similar trade-off derived from information-theoretic considerations has been shown to produce state-of-the-art results in practice (Slonim et al., 2005; Yom-Tov and Slonim, 2009). This paper supports the empirical evidence by providing a better theoretical foundation, suggesting formal generalization guarantees, and offering a more accurate way to deal with finite sample issues.

2. PAC-Bayesian Generalization Bounds

This section is devoted to PAC-Bayesian generalization bounds, which are the main tool used for the analysis of our learning models in the subsequent sections. We review the well-known PAC-Bayesian bound for classification and present a slight variation of a less well-known PAC-Bayesian bound for discrete density estimation. The PAC-Bayesian generalization bounds pioneered by McAllester (1998, 1999) provide guarantees on generalization abilities of randomized predictors (formally defined below in Section 2.1) within the classical PAC learning model (Valiant, 1984)

and build upon preceding works on PAC analysis of Bayesian learning models (Shawe-Taylor et al., 1998; Shawe-Taylor and Williamson, 1997). The classical PAC learning model evaluates learning algorithms by their ability to predict new events generated by the same probability distribution as the one that was used to train the algorithm. No restrictions on the data generating probability distribution are imposed except the assumption that the samples are i.i.d. The PAC-Bayesian framework should be distinguished from Bayesian learning, which assumes that the data were generated by hypotheses from the hypothesis class and applies Bayes' rule for inference. Bayesian learning does not provide guarantees on the expected error of the Bayes' inference rule and in some situations can lead to overfitting (Kearns et al., 1997).

The classical PAC bounds are derived by covering the error space of a hypothesis class. For example, the most familiar PAC bounds are based on the VC-dimension of a hypothesis class, which is a logarithm of the maximal number of points that can be jointly classified in any possible way by functions from the hypothesis class (Vapnik and Chervonenkis, 1968, 1971; Vapnik, 1998; Devroye et al., 1996). More recent bounds involve Rademacher and Gaussian complexities (Koltchinskii, 2001; Bartlett et al., 2001; Bartlett and Mendelson, 2001; Boucheron et al., 2005). However, in all the above approaches the whole hypothesis class is characterized by a single number: its VC-dimension or Rademacher complexity, which means that all the incorporating hypotheses are treated identically and there is no way to differentiate them and give preference to "simpler" ones. For example, if the hypothesis class consists of straight lines and parabolas, its VC-dimension is equal to the VC-dimension of a hypothesis class consisting of parabolas only and there is no direct way to give preference to straight lines within the combined hypothesis class. PAC-Bayesian bounds are derived by covering the hypothesis space and they enable non-uniform treatment of the hypotheses. In the PAC-Bayesian approach each hypothesis is characterized by its own complexity defined by its prior. This refined approach provides several important benefits, which include: (1) the ability to give explicit preference to certain hypotheses (e.g, in the example above we can assign a higher prior to straight lines); (2) a gradient within the hypothesis space, which can be used in algorithms for bound minimization; (3) considerably tighter bounds, which are meaningful in a practical sense: in some applications the discrepancy between the bound value and the test error is only 10%-20% (Langford, 2005; Seldin and Tishby, 2008; Seldin, 2009; Germain et al., 2009). There is one more distinction between the usual PAC analysis and the PAC-Bayesian bounds that extend the scope of applicability of the latter. Classical PAC analysis aims at bounding the discrepancy between the expected performance of the hypothesis with the best empirical performance and the best hypothesis within the hypothesis class. Such types of bounds require a uniform bound on the discrepancies between empirical and expected performances for all the hypotheses within a hypothesis class. The uniform bound exists if and only if the hypothesis class has a finite VC-dimension. PAC-Bayesian bounds bound the expected performance of a given hypothesis, but do not attempt to bound its gap to the performance of the best hypothesis within the hypothesis class. This fact makes it possible to apply PAC-Bayesian bounds even in situations where the VC-dimension of a hypothesis class is infinite, for example, decision trees of unlimited depth or separating hyperplanes in infinite-dimensional spaces. This does not disprove the fundamental theorem of PAC learning theory, which states that learning is possible if and only if the VC-dimension of a hypothesis class is finite, but rather extends the notion of learnability. Instead of a regret-based definition of learnability, by which the ability to learn is the ability to achieve, up to a small epsilon, the best possible solution within a hypothesis class, the PAC-Bayesian approach defines learnability as the ability to bound the

expected performance of the obtained solution. Then it is a question for the user to decide whether the guaranteed expected performance is sufficient for his or her needs.

Since the strength of PAC-Bayesian analysis lies in its ability to provide a non-uniform treatment of the hypotheses within a hypothesis class, its advantage over traditional PAC analysis is best seen in the analysis of heterogeneous hypothesis classes (or, in other words, when the hypotheses constituting a hypothesis class are not symmetric). Some hypothesis classes exhibit a "natural heterogeneity"; for example, we can partition the class of decision trees into subclasses according to tree depth. A higher prior can then be assigned to shallow trees to provide them with preference over deep trees. For example, a prior $\mathcal{P}(t) = 2^{-(d(t)+1)}2^{-2^{d(t)}}$, where $d(t)$ is the depth of a tree t would be a legal prior over the space of full binary decision trees of unlimited depth ($2^{-(d(t)+1)}$ is a prior over tree depth and $2^{-2^{d(t)}}$ is a prior over trees of a given depth). Note that it is possible to assign a higher prior to *all* shallow trees simultaneously because there are fewer shallow trees than deep trees. Hence, the above prior exploits knowledge about the structure of the hypothesis space, but makes no assumptions about the data. As we will see below, the bounds depend on $-\ln \mathcal{P}(t) = \ln(2)[(d(t)+1) + 2^{d(t)}]$; thus the value of the logarithm of the first part of the prior $(d(t)+1)$, which accounts for the tree depth, is negligible compared to the value of the logarithm of the second half of the prior $(2^{d(t)})$, which counts the number of symmetric trees given the depth. Given a strong prior knowledge on the problem domain, which breaks the symmetry between trees of a given depth, it is possible to give preference (a higher prior) to certain deep trees; however it is impossible to give a higher prior to all deep trees simultaneously, because there are too many of them. Thus, a choice of a different prior over tree depth and even precise prior knowledge of the tree depth can only negligibly improve the bound.

For some hypothesis spaces which seem homogeneous at a first glance it may still be possible to identify non-trivial asymmetries and define a corresponding structural prior. The best example comes from the analysis of SVMs, where the class of all possible separating hyperplanes in $\mathbb{R}^d$ is partitioned into subclasses according to the size of the margin and a higher prior is given to the hyperplanes with large margins (Langford and Shawe-Taylor, 2002; McAllester, 2003; Langford, 2005). In structure learning the hypothesis class usually exhibits a natural heterogeneity since the hypotheses (structures) can be differentiated by their complexity. Hence, PAC-Bayesian analysis has great potential in the analysis of structure learning which is only partially explored in this work. PAC-Bayesian bounds are further distinguished by their explicit dependence on model parameters, which makes their optimization easy.

Following the pioneering work of McAllester (1998, 1999), the PAC-Bayesian bounds were tightened and simplified by Seeger (2002, 2003). Some further improvements were suggested in Maurer (2004), Audibert and Bousquet (2007) and Blanchard and Fleuret (2007). This section draws on the easier-to-read expositions by Maurer (2004) and Banerjee (2006). In order to present the bounds for classification and density estimation, we need to define the notion of randomized predictors, which is done next. We then present the PAC-Bayesian theorems and their proofs.

2.1 Randomized Predictors

Let $\mathcal{H}$ be a hypothesis class and let $\mathcal{Q}(h)$ be a distribution over $\mathcal{H}$ (if $\mathcal{H}$ is infinite then $\mathcal{Q}(h)$ is a probability density). A *randomized predictor* associated with $\mathcal{Q}$, and with a small abuse of notation denoted by $\mathcal{Q}$, is defined in the following way: For each sample x a hypothesis $h \in \mathcal{H}$ is drawn according to $\mathcal{Q}(h)$, and then applied to make a prediction on x. In the classification context, $\mathcal{Q}$

is termed a *randomized classifier* (Langford, 2005). However, since this work extends the PAC-Bayesian framework beyond the classification scenario by using the same randomization technique, we use the term "randomized predictor". In this more general context $h(x)$ is a general function of x, not necessarily a classifier.

In the context of classification, let $S = \{(x_1, y_1), .., (x_N, y_N)\}$ be an i.i.d. sample of size N of instances and their labels drawn according to unknown distribution $p(x, y)$ and let $l(y, y')$ be a given loss function for predicting y' instead of y. Let $h(x)$ be the label of x predicted by hypothesis h. For each $h \in \mathcal{H}$ we denote by $\hat{L}(h) = \frac{1}{N}\sum_i l(y_i, h(x_i))$ the empirical loss of the hypothesis h on S and by $L(h) = \mathbb{E}_{p(x,y)} l(Y, h(X))$ the expected loss of h with respect to the true, unknown distribution that generates the data. We further extend the definitions of the empirical and expected losses for randomized predictors in the following way:

$$\hat{L}(\mathcal{Q}) = \mathbb{E}_{\mathcal{Q}(h)}\hat{L}(h) \qquad \text{and} \qquad L(\mathcal{Q}) = \mathbb{E}_{\mathcal{Q}(h)}L(h).$$

For two distributions q and p over domain $\mathcal{X}$ we define

$$KL(q\|p) = \mathbb{E}_q \ln \frac{q(x)}{p(x)}$$

to be the Kullback-Leibler (KL) divergence between q and p (Cover and Thomas, 1991). As well, we define

$$kl(\hat{L}(\mathcal{Q})\|L(\mathcal{Q})) = \hat{L}(\mathcal{Q}) \ln \frac{\hat{L}(\mathcal{Q})}{L(\mathcal{Q})} + (1 - \hat{L}(\mathcal{Q})) \ln \frac{1 - \hat{L}(\mathcal{Q})}{1 - L(\mathcal{Q})}$$

as the KL-divergence between two Bernoulli distributions with biases $\hat{L}(\mathcal{Q})$ and $L(\mathcal{Q})$. Now we are ready to state the PAC-Bayesian theorems.

2.2 PAC-Bayesian Theorems

Theorem 1 (PAC-Bayesian bound for classification) *For a hypothesis class $\mathcal{H}$, a prior distribution $\mathcal{P}$ over $\mathcal{H}$ and a zero-one loss function l, with probability greater than $1 - \delta$ over drawing a sample of size N, for all randomized classifiers $\mathcal{Q}$ simultaneously*:

$$kl(\hat{L}(\mathcal{Q})\|L(\mathcal{Q})) \leq \frac{KL(\mathcal{Q}\|\mathcal{P}) + \ln(N+1) - \ln\delta}{N}. \tag{1}$$

Theorem 2 (PAC-Bayesian bound for discrete density estimation) *Let $\mathcal{X}$ be the sample space (possibly infinite) and let $p(x)$ be an unknown distribution over $x \in \mathcal{X}$. Let $\mathcal{H}$ be a hypothesis class, such that each member $h \in \mathcal{H}$ is a function from $\mathcal{X}$ to a finite set $\mathcal{Z}$ with cardinality $|\mathcal{Z}|$. Let $p_h(z) = P_{X\sim p(x)}\{h(X) = z\}$ be the distribution over $\mathcal{Z}$ induced by $p(x)$ and h. Let $\mathcal{P}$ be a prior distribution over $\mathcal{H}$. Let $\mathcal{Q}$ be an arbitrary distribution over $\mathcal{H}$ and $p_{\mathcal{Q}}(z) = \mathbb{E}_{\mathcal{Q}(h)} p_h(z)$ a distribution over $\mathcal{Z}$ induced by $p(x)$ and $\mathcal{Q}$. Let S be an i.i.d. sample of size N generated according to $p(x)$ and let $\hat{p}(x)$ be the empirical distribution over $\mathcal{X}$ corresponding to S. Let $\hat{p}_h(z) = P_{X\sim\hat{p}(x)}\{h(X) = z\}$ be the empirical distribution over $\mathcal{Z}$ corresponding to h and S. Let $\hat{p}_{\mathcal{Q}}(z) = \mathbb{E}_{\mathcal{Q}(h)}\hat{p}_h(z)$. Then with probability greater than $1 - \delta$ for all possible $\mathcal{Q}$ simultaneously*:

$$KL(\hat{p}_{\mathcal{Q}}(z)\|p_{\mathcal{Q}}(z)) \leq \frac{KL(\mathcal{Q}\|\mathcal{P}) + (|\mathcal{Z}| - 1)\ln(N+1) - \ln\delta}{N}. \tag{2}$$

Remarks:

1. This form of Theorem 1 first appeared in Seeger (2002). A slightly different version of Theorem 2 first appeared in Seeger (2003) and independently in Seldin and Tishby (2009), where it found the first non-trivial application.

2. The PAC-Bayesian bound for classification (1) is a direct consequence of the PAC-Bayesian bound for density estimation (2). To see this, let Z be the error variable. Then each hypothesis $h \in \mathcal{H}$ is a function from the sample space (in this case the samples are pairs $\langle X, Y \rangle$) to the error variable Z and $|\mathcal{Z}| = 2$. Furthermore, $\hat{L}(h) = \hat{p}_h(Z = 1)$ and $L(h) = p_h(Z = 1)$, hence $kl(\hat{L}(\mathcal{Q}) \| L(\mathcal{Q})) = KL(\hat{p}_{\mathcal{Q}}(z) \| p_{\mathcal{Q}}(z))$. Substituting this into (2) yields (1).

3. Maurer (2004) showed that due to convexity of the KL-divergence, inequality (1) is valid for all loss functions bounded in the [0,1] interval, and not only for the zero-one loss. He also proved that due to tighter concentration of empirical means of binary variables, for $N \geq 8$ bound (1) can be further tightened:

$$kl(\hat{L}(\mathcal{Q}) \| L(\mathcal{Q})) \leq \frac{KL(\mathcal{Q} \| \mathcal{P}) + \frac{1}{2} \ln(4N) - \ln \delta}{N} \tag{3}$$

   and that this is the tightest result that can be proved using the techniques which are also used in this paper.

4. Although there is no analytical expression for the inverse of the kl-divergence, $L(\mathcal{Q})$ can be bounded numerically:

$$\begin{aligned} L(\mathcal{Q}) &\leq kl^{-1}\left(\hat{L}(\mathcal{Q}), \frac{KL(\mathcal{Q} \| \mathcal{P}) + \frac{1}{2} \ln(4N) - \ln \delta}{N}\right) \\ &= \max\left\{v : kl(\hat{L}(\mathcal{Q}) \| v) \leq \frac{KL(\mathcal{Q} \| \mathcal{P}) + \frac{1}{2} \ln(4N) - \ln \delta}{N}\right\}. \end{aligned} \tag{4}$$

   Since $kl(\hat{L}(\mathcal{Q}) \| v)$ is convex in v, (4) is easy to compute.

5. The proof of Theorem 2 presented below reveals a close relation between the PAC-Bayesian theorems and the method of types in information theory (Cover and Thomas, 1991). The trade-off between $\hat{L}(\mathcal{Q})$ and $KL(\mathcal{Q} \| \mathcal{P})$ in the PAC-Bayesian bounds also has a tight relation to the maximum entropy principle in learning and statistical mechanics (Jaynes, 1957; Dudík et al., 2007; Catoni, 2007; Shawe-Taylor and Hardoon, 2009). The relations between the PAC-Bayesian bounds, information theory, and statistical mechanics are further discussed in Catoni (2007).

The proof of Theorem 2 presented below is based on two auxiliary results which have value in their own right and therefore are presented in dedicated subsections. The first auxiliary result applies the method of types to bound the expectation of the exponent of the divergence between empirical and expected distributions over $\mathcal{Z}$ for a single hypothesis: $\mathbb{E}_S e^{N \cdot KL(\hat{p}_h(z) \| p_h(z))}$. The second auxiliary result relates the divergence $\mathbb{E}_{\mathcal{Q}(h)} KL(\hat{p}_h(z) \| p_h(z))$ for all $\mathcal{Q}$ to a single (prior) reference measure $\mathcal{P}$. This relation is actually the cornerstone of the PAC-Bayesian analysis. Finally, in Section 2.5 a quantity depending on the prior measure $\mathcal{P}$ is treated using the first auxiliary result to obtain the final bound in Equation (2).

2.3 The Law of Large Numbers

In this subsection we analyze the rate of convergence of empirical distributions over finite domains to their true values. The following result is based on the method of types in information theory (Cover and Thomas, 1991).

Theorem 3 *Let $S = \{X_1, .., X_N\}$ be i.i.d. distributed by $p(x)$. Denote by $\hat{p}(x)$ the empirical distribution over $\mathcal{X}$ corresponding to S and by $|\mathcal{X}|$ the cardinality of $\mathcal{X}$. Then*:

$$\mathbb{E}_S e^{N \cdot KL(\hat{p}(x) \| p(x))} \leq (N+1)^{|\mathcal{X}|-1}. \tag{5}$$

Proof Enumerate the possible values of $\mathcal{X}$ by $1, .., |\mathcal{X}|$ and let n_i count the number of occurrences of value i. Let p_i denote the probability of value i and $\hat{p}_i = \frac{n_i}{N}$ be its empirical counterpart. Let $H(\hat{p}) = -\sum_i \hat{p}_i \ln \hat{p}_i$ be the empirical entropy. Then:

$$\begin{aligned}
\mathbb{E}_S e^{NKL(\hat{p}\|p)} &= \sum_{\substack{n_1,..,n_{|\mathcal{X}|}: \\ \sum_i n_i = N}} \binom{N}{n_1, .., n_{|\mathcal{X}|}} \cdot \prod_{i=1}^{|\mathcal{X}|} p_i^{N\hat{p}_i} \cdot e^{NKL(\hat{p}\|p)} \\
&\leq \sum_{\substack{n_1,..,n_{|\mathcal{X}|}: \\ \sum_i n_i = N}} e^{NH(\hat{p})} \cdot e^{N \sum_i \hat{p}_i \ln p_i} \cdot e^{NKL(\hat{p}\|p)} \qquad (6) \\
&= \sum_{\substack{n_1,..,n_{|\mathcal{X}|}: \\ \sum_i n_i = N}} 1 = \binom{N+|\mathcal{X}|-1}{|\mathcal{X}|-1} \leq (N+1)^{|\mathcal{X}|-1}. \qquad (7)
\end{aligned}$$

In (6) we used the $\binom{N}{n_1,..,n_{|\mathcal{X}|}} \leq e^{NH(\hat{p})}$ bound on the multinomial coefficient, which counts the number of sequences with a fixed cardinality profile (unnormalized type) $n_1, .., n_{|\mathcal{X}|}$ (Cover and Thomas, 1991). In the second equality in (7) the number of ways to choose n_i-s equals the number of ways we can place $|\mathcal{X}|-1$ ones in a sequence of $N+|\mathcal{X}|-1$ ones and zeros, where ones symbolize a partition of zeros ("balls") into $|\mathcal{X}|$ bins. ■

2.4 Change of Measure Inequality

The simultaneous treatment of all possible distributions (measures) $\mathcal{Q}$ over $\mathcal{H}$ is done by relating them all to a single reference (prior) measure $\mathcal{P}$. We call this relation a *change of measure inequality*. This inequality was formulated as a standalone result in Banerjee (2006), although it originates much earlier. Banerjee (2006) terms it a *compression lemma*; however we find the term "change of measure inequality" more appropriate to its nature and usage. The inequality is a simple consequence of Jensen's inequality.

Lemma 4 (Change of Measure Inequality) *For any measurable function $\phi(h)$ on $\mathcal{H}$ and any distributions $\mathcal{P}$ and $\mathcal{Q}$ on $\mathcal{H}$, we have:*

$$\mathbb{E}_{\mathcal{Q}(h)} \phi(h) \leq KL(\mathcal{Q}\|\mathcal{P}) + \ln \mathbb{E}_{\mathcal{P}(h)} e^{\phi(h)}. \tag{8}$$

Proof For any measurable function $\phi(h)$, we have:

$$\begin{aligned}
\mathbb{E}_{\mathcal{Q}(h)}\phi(h) &= \mathbb{E}_{\mathcal{Q}(h)} \ln \left(\frac{\mathcal{Q}(h)}{\mathcal{P}(h)} \cdot e^{\phi(h)} \cdot \frac{\mathcal{P}(h)}{\mathcal{Q}(h)} \right) \\
&= KL(\mathcal{Q}\|\mathcal{P}) + \mathbb{E}_{\mathcal{Q}(h)} \ln \left(e^{\phi(h)} \cdot \frac{\mathcal{P}(h)}{\mathcal{Q}(h)} \right) \\
&\leq KL(\mathcal{Q}\|\mathcal{P}) + \ln \mathbb{E}_{\mathcal{Q}(h)} \left(e^{\phi(h)} \cdot \frac{\mathcal{P}(h)}{\mathcal{Q}(h)} \right) \qquad (9) \\
&= KL(\mathcal{Q}\|\mathcal{P}) + \ln \mathbb{E}_{\mathcal{P}(h)} e^{\phi(h)},
\end{aligned}$$

where (9) is by Jensen's inequality. ■

2.5 Proof of the PAC-Bayesian Generalization Bound for Density Estimation

We apply the results of the previous two subsections to prove the PAC-Bayesian generalization bound for density estimation.

Proof of Theorem 2 Let $\phi(h, S, p) = NKL(\hat{p}_h(z)\|p_h(z))$. Then:

$$\begin{aligned}
NKL(\hat{p}_{\mathcal{Q}}(z)\|p_{\mathcal{Q}}(z)) &= NKL(\mathbb{E}_{\mathcal{Q}(h)}\hat{p}_h(z)\|\mathbb{E}_{\mathcal{Q}(h)}p_h(z)) \\
&\leq \mathbb{E}_{\mathcal{Q}(h)} NKL(\hat{p}_h(z)\|p_h(z)) \qquad (10) \\
&\leq KL(\mathcal{Q}\|\mathcal{P}) + \ln \mathbb{E}_{\mathcal{P}(h)} e^{NKL(\hat{p}_h(z)\|p_h(z))}, \qquad (11)
\end{aligned}$$

where (10) is by the convexity of the KL-divergence (Cover and Thomas, 1991) and (11) is by the change of measure inequality. To obtain (2) it is left to bound $\mathbb{E}_{\mathcal{P}(h)} e^{NKL(\hat{p}_h(z)\|p_h(z))}$. This is a random quantity depending on the sample S since $\hat{p}_h(z)$ for each h depends on the sample. By Markov's inequality we know that with probability at least $1-\delta$ over the sample $\mathbb{E}_{\mathcal{P}(h)} e^{NKL(\hat{p}_h(z)\|p_h(z))} \leq \frac{1}{\delta}\mathbb{E}_S \left[\mathbb{E}_{\mathcal{P}(h)} e^{NKL(\hat{p}_h(z)\|p_h(z))} \right]$. In order to obtain a bound on $\mathbb{E}_S \left[\mathbb{E}_{\mathcal{P}(h)} e^{NKL(\hat{p}_h(z)\|p_h(z))} \right]$ we note that it is possible to exchange $\mathbb{E}_S$ with $\mathbb{E}_{\mathcal{P}(h)}$ since S and h are independent:

$$\mathbb{E}_S \left[\mathbb{E}_{\mathcal{P}(h)} e^{NKL(\hat{p}_h(z)\|p_h(z))} \right] = \mathbb{E}_{\mathcal{P}(h)} \left[\mathbb{E}_S e^{NKL(\hat{p}_h(z)\|p_h(z))} \right] \leq (N+1)^{|\mathcal{Z}|-1}. \qquad (12)$$

The last inequality in (12) is justified by the fact that $\mathbb{E}_S e^{NKL(\hat{p}_h(z)\|p_h(z))} \leq (N+1)^{|\mathcal{Z}|-1}$ for each h individually according to (5). By Markov's inequality we conclude that with probability of at least $1-\delta$ over S:

$$\mathbb{E}_{\mathcal{P}(h)} e^{NKL(\hat{p}_h(z)\|p_h(z))} \leq \frac{(N+1)^{|\mathcal{Z}|-1}}{\delta}.$$

Substituting this into (11) and normalizing by N yields (2). ■

2.6 Addendum to the Law of Large Numbers

Note in passing that it is straightforward to recover theorem 12.2.1 in Cover and Thomas (1991) from Theorem 3 (even with a slight improvement). This theorem is used later in the estimation of marginal distributions.

Theorem 5 (12.2.1 in Cover and Thomas, 1991) *Under the notations of Theorem 3 with probability greater than* $1-\delta$:

$$KL(\hat{p}(x)\|p(x)) \leq \frac{(|\mathcal{X}|-1)\ln(N+1)-\ln\delta}{N}.$$

Proof Immediate from application of Markov's inequality to (5). ■

2.7 Construction of a Density Estimator

Although we have bounded $KL(\hat{p}_{\mathcal{Q}}(z)\|p_{\mathcal{Q}}(z))$ in Theorem 2, $\hat{p}_{\mathcal{Q}}(z)$ still cannot be used as a density estimator for $p_{\mathcal{Q}}(z)$, because it is not bounded from zero. In order to bound the logarithmic loss $-\mathbb{E}_{p_{\mathcal{Q}}(z)}\ln\hat{p}_{\mathcal{Q}}(z)$, which corresponds, for example, to the expected code length of encoder $\hat{p}_{\mathcal{Q}}$ when samples are generated by $p_{\mathcal{Q}}$, we have to smooth $\hat{p}_{\mathcal{Q}}$. We denote a smoothed version of $\hat{p}_{\mathcal{Q}}$ by $\tilde{p}_{\mathcal{Q}}$ and define it as:

$$\tilde{p}_h(z) = \frac{\hat{p}_h(z)+\gamma}{1+\gamma|\mathcal{Z}|}, \tag{13}$$
$$\tilde{p}_{\mathcal{Q}}(z) = \mathbb{E}_{\mathcal{Q}(h)}\tilde{p}_h(z) = \frac{\hat{p}_{\mathcal{Q}}(z)+\gamma}{1+\gamma|\mathcal{Z}|}.$$

In the following theorem we show that if $KL(\hat{p}_{\mathcal{Q}}(z)\|p_{\mathcal{Q}}(z)) \leq \varepsilon(\mathcal{Q})$ and $\gamma(\mathcal{Q}) = \frac{\sqrt{\varepsilon(\mathcal{Q})/2}}{|\mathcal{Z}|}$, then $-\mathbb{E}_{p_{\mathcal{Q}}(z)}\ln\tilde{p}_{\mathcal{Q}}(z)$ is roughly within $\pm\sqrt{\varepsilon(\mathcal{Q})/2}\ln|\mathcal{Z}|$ range around $H(\hat{p}_{\mathcal{Q}}(z))$. The bound on $KL(\hat{p}_{\mathcal{Q}}(z)\|p_{\mathcal{Q}}(z))$ is naturally obtained by Theorem 2. Thus, the performance of the density estimator $\tilde{p}_{\mathcal{Q}}$ is optimized by distribution $\mathcal{Q}$ that minimizes the trade-off between $H(\hat{p}_{\mathcal{Q}}(z))$ and $\frac{1}{N}KL(\mathcal{Q}\|\mathcal{P})$.

Note that for a uniform distribution $u(z) = \frac{1}{|\mathcal{Z}|}$ the value of $-\mathbb{E}_{p(z)}\ln u(z) = \ln|\mathcal{Z}|$. Thus, the theorem below is interesting when $\sqrt{\varepsilon(\mathcal{Q})/2}$ is significantly smaller than 1. For technical reasons in the proofs of the following section, the upper bound in the next theorem is stated for $-\mathbb{E}_{p_{\mathcal{Q}}(z)}\ln\tilde{p}_{\mathcal{Q}}(z)$ and for $-\mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{p_h(z)}\ln\tilde{p}_h(z)$. We also denote $\varepsilon = \varepsilon(\mathcal{Q})$ for brevity.

Theorem 6 *Let Z be a random variable distributed according to $p_{\mathcal{Q}}(z)$ and assume that $KL(\hat{p}_{\mathcal{Q}}(z)\|p_{\mathcal{Q}}(z)) \leq \varepsilon$. Then $-\mathbb{E}_{p_{\mathcal{Q}}(z)}\ln\tilde{p}_{\mathcal{Q}}(z)$ is minimized by $\gamma = \frac{\sqrt{\varepsilon/2}}{|\mathcal{Z}|}$. For this value of γ the following inequalities hold:*

$$-\mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{p_h(z)}\ln\tilde{p}_h(z) \leq H(\hat{p}_{\mathcal{Q}}(z)) + \sqrt{\varepsilon/2}\ln|\mathcal{Z}| + \phi(\varepsilon), \tag{14}$$
$$-\mathbb{E}_{p_{\mathcal{Q}}(z)}\ln\tilde{p}_{\mathcal{Q}}(z) \leq H(\hat{p}_{\mathcal{Q}}(z)) + \sqrt{\varepsilon/2}\ln|\mathcal{Z}| + \phi(\varepsilon), \tag{15}$$
$$-\mathbb{E}_{p_{\mathcal{Q}}(z)}\ln\tilde{p}_{\mathcal{Q}}(z) \geq H(\hat{p}_{\mathcal{Q}}(z)) - \sqrt{\varepsilon/2}\ln|\mathcal{Z}| - \psi(\varepsilon), \tag{16}$$

where:

$$\psi(\varepsilon) = \sqrt{\varepsilon/2}\ln\frac{1+\sqrt{\varepsilon/2}}{\sqrt{\varepsilon/2}} \;\; \textit{and} \;\; \phi(\varepsilon) = \psi(\varepsilon) + \ln(1+\sqrt{\varepsilon/2}).$$

The proof is provided in appendix A. Note that both $\phi(\varepsilon)$ and $\psi(\varepsilon)$ converge to zero approximately as $-\sqrt{\varepsilon/2}\ln\sqrt{\varepsilon/2}$ and for $\sqrt{\varepsilon/2} < \frac{1}{|\mathcal{Z}|}$ they are in fact dominant over the $\sqrt{\varepsilon/2}\ln|\mathcal{Z}|$ term. Nevertheless, the main message is still that in order to minimize $-\mathbb{E}_{p_{\mathcal{Q}}(z)}\ln\tilde{p}_{\mathcal{Q}}(z)$ the trade-off between $H(\hat{p}_{\mathcal{Q}}(z))$ and $KL(\hat{p}_{\mathcal{Q}}(z)\|p_{\mathcal{Q}}(z))$ should be minimized. This message is explored in more details in Section 3.4.

Remark: As an aside we consider the case of direct density estimation. Assume we are given a set of N i.i.d. observations $x_1,..,x_N$ generated according to an unknown distribution $p(x)$ over a finite domain $\mathcal{X}$. We want to construct an estimate $\tilde{p}(x)$ for $p(x)$ based on the empirical frequencies $\hat{p}(x)$, such that the expectation $-\mathbb{E}_{p(x)}\ln\tilde{p}(x)$ is minimized. This problem, known as "histogram smoothing", has received significant attention in statistics and information theory (Gilbert, 1971; Cover, 1972; Krichevskiy, 1998; Paninski, 2004). Uniform smoothing of the form

$$\tilde{p}(x) = \frac{\hat{p}(x)+\gamma}{1+\gamma|\mathcal{X}|},$$

such as the one applied in (13) is known as the Dirichlet-Bayes or "add-constant" estimator. Theorem 6 provides the optimal value of γ

$$\gamma = \frac{1}{|\mathcal{X}|}\sqrt{\varepsilon/2} = \frac{1}{|\mathcal{X}|}\sqrt{\frac{(|\mathcal{X}|-1)\ln(N+1)-\ln\delta}{2N}}$$

for which with probability greater than $1-\delta$ over the sample

$$H(\hat{p}(x)) - \sqrt{\varepsilon/2}\ln|\mathcal{X}| - \psi(\varepsilon) \le -\mathbb{E}_{p(x)}\ln\tilde{p}(x) \le H(\hat{p}(x)) + \sqrt{\varepsilon/2}\ln|\mathcal{X}| + \phi(\varepsilon),$$

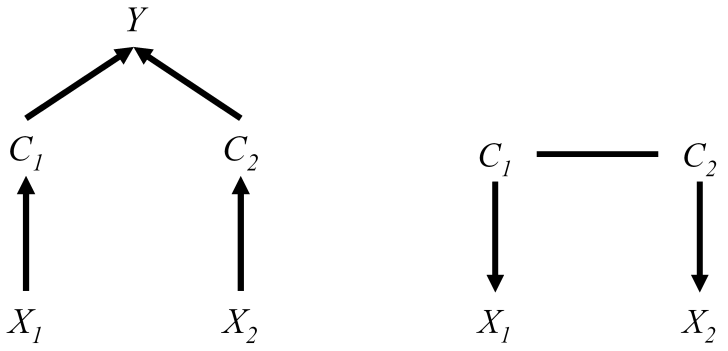
where $\varepsilon = \frac{(|\mathcal{X}|-1)\ln(N+1)-\ln\delta}{N}$ is obtained from Theorem 5. By Theorem 6 the optimal smoothing γ decreases as the sample size N increases. A more detailed comparison of this result with preceding work is beyond the scope of this paper and will be presented elsewhere. Note that in the more general case considered in Theorem 6, where the distribution $p_{\mathcal{Q}}(z)$ depends on $\mathcal{Q}$ the smoothing parameter γ also depends on $\mathcal{Q}$.

3. PAC-Bayesian Analysis of Co-clustering

In the introduction we defined two high-level goals, which can be solved via co-clustering. The first is discriminative prediction of the matrix entries, as in the collaborative filtering example. The second is estimation of the joint probability distribution in co-occurrence data analysis. We further defined the notion of generalization for each of the two problems. In this section we derive PAC-Bayesian generalization bounds for the two settings. We begin with the co-clustering approach to discriminative prediction, which is slightly easier in terms of presentation. Then we consider the discrete density estimation problem.

3.1 PAC-Bayesian Analysis of Discriminative Prediction with Grid Clustering

Let $\mathcal{X}_1 \times .. \times \mathcal{X}_d \times \mathcal{Y}$ be a $(d+1)$-dimensional product space. We assume that each $\mathcal{X}_i$ is categorical and its cardinality, denoted by $|\mathcal{X}_i| = n_i$, is fixed and known. We also assume that $\mathcal{Y}$ is finite with cardinality $|\mathcal{Y}|$ and that a bounded loss function $l(y,y')$ for predicting y' instead of y is given. As an example, consider collaborative filtering. In collaborative filtering $d=2$, $\mathcal{X}_1$ is the space of the



(a) Graphical Model for Discriminative Prediction

(b) Graphical Model for Density Estimation

Figure 1: Illustration of graphical models corresponding to discriminative prediction (17) and density estimation (26). The illustrations are for $d = 2$.

viewers, n_1 is the number of viewers, $\mathcal{X}_2$ is the space of the movies, n_2 is the number of movies, and $\mathcal{Y}$ is the space of the ratings (e.g., on a five-star scale). The loss $l(y, y')$ can be, for example, the absolute loss $l(y, y') = |y - y'|$ or the quadratic loss $l(y, y') = (y - y')^2$. There is no natural metric either on the space of viewers or on the space of movies; thus both $\mathcal{X}_1$ and $\mathcal{X}_2$ are categorical.

We assume an existence of an unknown probability distribution $p(x_1, .., x_d, y)$ over the $\mathcal{X}_1 \times .. \times \mathcal{X}_d \times \mathcal{Y}$ product space. We further assume that we are given an i.i.d. sample of size N generated according to $p(x_1, .., x_d, y)$. We use $\hat{p}(x_1, .., x_d, y)$ to denote the empirical frequencies of $(d+1)$-tuples $\langle x_1, .., x_d, y \rangle$ in the sample. We consider the following form of discriminative predictors:

$$q(y|x_1, .., x_d) = \sum_{c_1,..,c_d} q(y|c_1, .., c_d) \prod_{i=1}^{d} q(c_i|x_i). \tag{17}$$

The hidden variables $C_1, .., C_d$ represent a clustering of the observed variables $X_1, .., X_d$. The hidden variable C_i accepts values in $\{1, .., m_i\}$, where $m_i = |C_i|$ denotes the number of clusters used along dimension i. The conditional probability distribution $q(c_i|x_i)$ represents the probability of mapping (assigning) x_i to cluster c_i. The conditional probability $q(y|c_1, .., c_d)$ represents the probability of assigning label y to cell $\langle c_1, .., c_d \rangle$ in the cluster product space. The prediction model (17) corresponds to the graphical model in Figure 1.a. Note that this is a two-level randomized prediction model. The free parameters of the model are the conditional distributions $\{q(c_i|x_i)\}_{i=1}^{d}$ and $q(y|c_1, .., c_d)$. We denote these collectively by $\mathcal{Q} = \left\{ \{q(c_i|x_i)\}_{i=1}^{d}, q(y|c_1, .., c_d) \right\}$. In the next subsection we show that (17) corresponds to a randomized prediction strategy. We further denote:

$$L(\mathcal{Q}) = \mathbb{E}_{p(x_1,..,x_d,y)} \mathbb{E}_{q(y'|x_1,..,x_d)} l(Y, Y')$$

and

$$\hat{L}(\mathcal{Q}) = \mathbb{E}_{\hat{p}(x_1,..,x_d,y)} \mathbb{E}_{q(y'|x_1,..,x_d)} l(Y, Y'),$$

where $q(y|x_1, .., x_d)$ is defined by (17). We define

$$\bar{I}(X_i; C_i) = \frac{1}{n_i} \sum_{x_i, c_i} q(c_i|x_i) \ln \frac{q(c_i|x_i)}{\bar{q}(c_i)},$$

where $x_i \in \mathcal{X}_i$ are the possible values of X_i, $c_i \in \{1,..,m_i\}$ are the possible values of C_i, and

$$\bar{q}(c_i) = \frac{1}{n_i} \sum_{x_i} q(c_i|x_i)$$

is the marginal distribution over C_i corresponding to $q(c_i|x_i)$ and a *uniform* distribution $u(x_i) = \frac{1}{n_i}$ over $\mathcal{X}_i$. Thus, $\bar{I}(X_i;C_i)$ is the mutual information corresponding to the joint distribution $\bar{q}(x_i,c_i) = \frac{1}{n_i} q(c_i|x_i)$ defined by $q(c_i|x_i)$ and the uniform distribution over $\mathcal{X}_i$.

With the above definitions we can state the following generalization bound for discriminative prediction with co-clustering.

Theorem 7 *For any probability measure $p(x_1,..,x_d,y)$ over $\mathcal{X}_1 \times .. \times \mathcal{X}_d \times \mathcal{Y}$ and for any loss function l bounded by 1, with probability of at least $1-\delta$ over a selection of an i.i.d. sample S of size N according to p, for all randomized classifiers $\mathcal{Q} = \left\{\{q(c_i|x_i)\}_{i=1}^d, q(y|c_1,..,c_d)\right\}$:*

$$kl(\hat{L}(\mathcal{Q}) \| L(\mathcal{Q})) \leq \frac{\sum_{i=1}^d \left(n_i \bar{I}(X_i;C_i) + m_i \ln n_i\right) + M \ln|\mathcal{Y}| + \frac{1}{2}\ln(4N) - \ln\delta}{N}, \qquad (18)$$

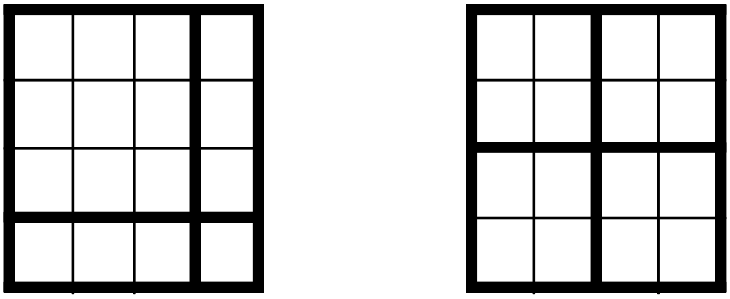
where M is the number of partition cells:

$$M = \prod_{i=1}^d m_i.$$

Remarks: Of course, any bounded loss can be normalized to the [0,1] interval. Note that given a prediction strategy $\mathcal{Q} = \left\{\{q(c_i|x_i)\}_{i=1}^d, q(y|c_1,..,c_d)\right\}$ both $\hat{L}(\mathcal{Q})$ and $\bar{I}(X_i;C_i)$ are computable exactly. $L(\mathcal{Q})$ can be bounded by numerical inversion of kl, as shown in Equation (4). To minimize $L(\mathcal{Q})$ both $\hat{L}(\mathcal{Q})$ and $\bar{I}(X_i;C_i)$ should be minimized.

Discussion: There are two extreme solutions to the collaborative filtering task that provide good intuitions on the co-clustering approach to this problem. If we assign all of the data to a single large cluster, we can evaluate the empirical mean/median/most frequent rating of that cluster fairly well. In this situation the empirical loss $\hat{L}(\mathcal{Q})$ is expected to be large, because we approximate all the entries with the global average, but its distance to the true loss $L(\mathcal{Q})$ is expected to be small. If we take the other extreme and assign each row and each column to a separate cluster, $\hat{L}(\mathcal{Q})$ can be zero given that we can approximate every entry with its own value, but its distance to the true loss $L(\mathcal{Q})$ is expected to be large because each cluster has too little data to make a statistically reliable estimation. Thus, the goal is to optimize the trade-off between the locality of the predictions and their statistical reliability.

This trade-off is explicitly exhibited in bound (18): if we assign all x_i-es to a single cluster, then $\bar{I}(X_i;C_i) = 0$ and therefore $L(\mathcal{Q})$ is close to $\hat{L}(\mathcal{Q})$. And if we assign each x_i to a separate cluster, then $\bar{I}(X_i;C_i)$ is large, specifically in this case $\bar{I}(X_i;C_i) = \ln n_i$, and $L(\mathcal{Q})$ is far from $\hat{L}(\mathcal{Q})$. But there are even finer observations we can draw from the bound. Bear in mind that $n_i \bar{I}(X_i;C_i)$ is linear in n_i, whereas $m_i \ln n_i$ is logarithmic in n_i. Thus, at least when m_i is small compared to n_i (which is a reasonable assumption when we cluster the values of $\mathcal{X}_i$) the leading term in (18) is $n_i \bar{I}(X_i;C_i)$. This term penalizes the *effective* complexity of a partition, rather than the raw number of clusters used. For example, the unbalanced partition of a 4×4 matrix into 2×2 clusters in Figure 2.a is simpler than the balanced partition into the same number of clusters in Figure 2.b. The reason,



(a) Unbalanced Partition (b) Balanced Partition

Figure 2: Illustration of (a) an unbalanced and (b) a balanced partition of a 4×4 matrix into 2×2 clusters. Note that there are 4 possible ways to group 4 objects into 2 unbalanced clusters and $\binom{4}{2} = 6$ possible ways to group 4 objects into 2 balanced clusters. Thus, the subspace of the unbalanced partitions is smaller than the subspace of the balanced partitions and the unbalanced partitions are simpler (it is easier to describe an unbalanced partition than a balanced one).

which will become clearer after we have defined the prior over the space of partitions in Section 3.3, is that there are fewer unbalanced partitions than balanced ones. Therefore, the subspace of unbalanced partitions is smaller than the subspace of balanced partitions and it is easier to describe an unbalanced partition than a balanced one. Intuitively, the partition in Figure 2.a does not fully use the 2×2 clusters that it could use, and should therefore be penalized less. On a practical level, the bound makes it possible to operate at the optimization step with more clusters than are actually required and to penalize the final solution according to a de facto measure of cluster use. This claim is supported by our experiments. To summarize this point, the bound (18) suggests a trade-off between the empirical performance and the effective complexity of a partition.

Finally, consider the $M \ln |\mathcal{Y}|$ term in the bound. M is the number of partition cells (in a hard partition) and $M \ln |\mathcal{Y}|$ corresponds to the size of the $\langle C_1, .., C_d, Y \rangle$ clique in the moral graph corresponding to the graph in Figure 1.a. The number of sample points N should be comparable to the number of partition cells, so it is natural for this term to appear in the bound. This term grows exponentially with the number of dimensions d; thus we can apply the bound for low-dimensional problems like collaborative filtering, but when the number of dimensions grows a different approach is required. We suggest one possible way to handle high dimensional problems in Section 7.

Proof of Theorem 7 The proof is a direct application of the PAC-Bayesian bound for classification in Theorem 1 (or, more precisely, its refinement in (3)). In order to apply the theorem, we have to define a hypothesis space $\mathcal{H}$, a prior over hypothesis space $\mathcal{P}$, a posterior over hypothesis space $\mathcal{Q}$, and calculate the KL-divergence $KL(\mathcal{Q}\|\mathcal{P})$. We define the hypothesis space in the next subsection and design a prior over it in Section 3.3. Substitution of the calculation of $KL(\mathcal{Q}\|\mathcal{P})$ in Lemma 9 into Theorem 1 completes the proof. ■

3.2 Grid Clustering Hypothesis Space

We define the hypothesis space $\mathcal{H}$ to be the space of hard grid partitions of the product space $\mathcal{X}_1 \times .. \times \mathcal{X}_d$ (as illustrated in Figure 2) augmented with label assignments to the partition cells. (In Section 3.4 we use grid partitions without labels on the partition cells; thus the discussion in this and the following subsection is kept general enough to hold in both cases.) In a hard grid partition each value $x_i \in \mathcal{X}_i$ is mapped deterministically to a single cluster $c_i \in \{1, .., m_i\}$. To operate on $\mathcal{H}$ we use the following notations:

- Let $\bar{m} = (m_1, .., m_d)$ be a vector counting the number of clusters along each dimension.
- We use $\mathcal{H}|_i$ to denote the space of partitions of $\mathcal{X}_i$. In other words, $\mathcal{H}|_i$ is a projection of $\mathcal{H}$ onto dimension i.
- Let $\mathcal{H}_{\bar{m}}$ denote the subspace of partitions of $\mathcal{X}_1 \times .. \times \mathcal{X}_d$ in which the number of clusters used along each dimension matches $\bar{m}$. Obviously, for distinct $\bar{m}$-s, $\mathcal{H}_{\bar{m}}$-s are disjoint.
- We use $\mathcal{H}|_{(y|\bar{m})}$ or simply $\mathcal{H}|_{y|\bar{m}}$ to denote the space of possible assignments of labels to $\mathcal{H}_{\bar{m}}$. Then we can write $\mathcal{H} = \bigcup_{\bar{m}} \left(\mathcal{H}_{\bar{m}} \times \mathcal{H}|_{y|\bar{m}} \right)$.
- For each $h \in \mathcal{H}$ we write $h = h|_1 \times .. \times h|_d \times h|_{y|\bar{m}}$, where $h|_i$ denotes the partition induced by h along dimension i and $h|_{y|\bar{m}}$ denotes the assignment of labels to partition cells of h. In the discussion of density estimation with grid clustering in Section 3.4, h is just $h = h|_1 \times .. \times h|_d$, without the labels assignment.

We show that $\mathcal{Q} = \left\{ \{q(c_i|x_i)\}_{i=1}^d, q(y|c_1, .., c_d) \right\}$ is a distribution over $\mathcal{H}$ and (17) corresponds to a randomized prediction strategy. More precisely, $\mathcal{Q}$ is a distribution over $\mathcal{H}_{\bar{m}} \times \mathcal{H}|_{y|\bar{m}}$, where $\bar{m}$ matches the cardinalities of C_i-s in the definitions of $\left\{ \{q(c_i|x_i)\}_{i=1}^d, q(y|c_1, .., c_d) \right\}$. In order to draw a hypothesis $h \in \mathcal{H}$ according to $\mathcal{Q}$ we draw a cluster c_i for each $x_i \in \mathcal{X}_i$ according to $q(c_i|x_i)$ and then draw a label for each partition cell according to $q(y|c_1, .., c_d)$. For example, we map each viewer to a cluster of viewers, map each movie to a cluster of movies, and assign ratings to the product space of viewer clusters by movie clusters. Then, in order to assign a label to a sample $\langle x_1, .., x_d \rangle$ we simply check which partition cell it has fallen into and return the corresponding label. Recall that in order to assign a label to another sample point, we have to draw a new hypothesis from $\mathcal{H}$.

Note that in (17) we actually skip the step of assigning a cluster for each $x_i \in \mathcal{X}_i$ and a label for each partition cell (in fact, the whole step of drawing a hypothesis) and assign a label to a given point $\langle x_1, .., x_d \rangle$ directly. Nevertheless, (17) corresponds to the randomized prediction process described above. This makes it possible to apply the PAC-Bayesian analysis.

3.3 Combinatorial Priors in PAC-Bayesian Bounds

In this section we design a combinatorial prior over the grid clustering hypothesis space and calculate the KL-divergence $KL(\mathcal{Q}\|\mathcal{P})$ between the posterior defined earlier and the prior. An interesting point about this result is that combinatorial priors result in mutual information terms in the calculations of the KL-divergence. This can be contrasted with the L_2-norm and L_1-norm terms resulting from Gaussian and Laplacian priors respectively in the analysis of SVMs (Langford, 2005). Another

important point to mention is that the posterior $\mathcal{Q}$ returns a named partition of $\mathcal{X}_i$-s (the conditional distribution $q(c_i|x_i)$ specifies the "name" c_i of the cluster that x_i is mapped to). However, the hypothesis space $\mathcal{H}$ and the prior $\mathcal{P}$ defined below operate with unnamed partitions: they only depend on the structure of a partition (the sizes of the clusters), but do not depend on the names assigned to the clusters. In this manner we account for all possible permutations of cluster names, which are irrelevant for the solution.

The statements in the next two lemmas are given in two versions, one for $\mathcal{H}$ augmented with labels, which is used in the proofs of Theorem 7, and the other for $\mathcal{H}_{\bar{m}}$ without the labels, which is used later for the proofs on density estimation with grid clustering.

Lemma 8 *It is possible to define a prior $\mathcal{P}$ over $\mathcal{H}_{\bar{m}}$ that satisfies*:

$$\mathcal{P}(h) \geq \frac{1}{\exp\left[\sum_{i=1}^{d}\left(n_i H(hist(h|_i)) + (m_i - 1)\ln n_i\right)\right]}, \tag{19}$$

where $hist(h|_i) = \{|c_{i1}|, .., |c_{im_i}|\}$ denotes the cardinality profile (histogram) of cluster sizes along dimension i of a partition corresponding to h and $H(hist(h|_i)) = -\sum_{j=1}^{m_i} \frac{|c_{ij}|}{n_i} \ln \frac{|c_{ij}|}{n_i}$ is the entropy of the (normalized) cardinality profile (note that $\sum_{j=1}^{m_i} |c_{ij}| = n_i$).

It is further possible to define a prior $\mathcal{P}$ over $\mathcal{H} = \bigcup_{\bar{m}} \left(\mathcal{H}_{\bar{m}} \times \mathcal{H}|_{y|\bar{m}}\right)$ that satisfies:

$$\mathcal{P}(h) \geq \frac{1}{\exp\left[\sum_{i=1}^{d}\left(n_i H(hist(h|_i)) + m_i \ln n_i\right) + M \ln |\mathcal{Y}|\right]}. \tag{20}$$

Remark: The prior $\mathcal{P}$ over $\mathcal{H}$ that is defined explicitly in the proof of the lemma exploits structural asymmetries between more and less balanced grid partitions, as shown in Figure 2, without making assumptions on the data generating process. Note that the leading terms in the prior are $n_i H(hist(h|_i))$ that count the number of possible ways to assign x_i-s to c_i-s, which are invariant under permutation of x_i-s within each $\mathcal{X}_i$ (see the proof for details). Thus, it is impossible to design a significantly better prior without "strong" prior knowledge on the data generating process that can break the permutation symmetry of x_i-s. "Weak" prior knowledge on the number of clusters m_i along each dimension and even on their sizes can only introduce an improvement that is logarithmic in n_i-s to the bounds. The PAC-Bayesian analysis enables us to operate with all possible grid partitions, while paying a very low (logarithmic) price for this generality.

We note that recently Lever et al. (2010) suggested an elegant technique for defining priors based on the true data distribution, which is, of course, a "strong" prior knowledge. For example, using the true marginal distributions $p(x_i)$ over $\mathcal{X}_i$-s in the definition of the prior would break the permutation symmetry of x_i-s. Since the true data distribution is independent of the sample, such priors are perfectly valid, although uncomputable. Lever et. al. show that, at least in some situations, the KL-divergence $KL(\mathcal{Q}\|\mathcal{P})$ can nevertheless be bounded, which is sufficient for application of the PAC-Bayesian bounds. The possibility of application of distribution-dependent priors in co-clustering will be explored in future work.

Lemma 9 *For the prior defined in* (19) *and posterior $\mathcal{Q} = \{q(c_i|x_i)\}_{i=1}^{d}$:*

$$KL(\mathcal{Q}\|\mathcal{P}) \leq \sum_{i=1}^{d}\left(n_i \bar{I}(X_i; C_i) + (m_i - 1)\ln n_i\right). \tag{21}$$

For the prior defined in (20) *and posterior* $\mathcal{Q} = \Big\{\{q(c_i|x_i)\}_{i=1}^d, q(y|c_1,..,c_d)\Big\}$:

$$KL(\mathcal{Q}\|\mathcal{P}) \leq \sum_{i=1}^{d}\Big(n_i\bar{I}(X_i;C_i) + m_i \ln n_i\Big) + M\ln|\mathcal{Y}|. \tag{22}$$

3.3.1 PROOFS

Proof of Lemma 8 To define the prior $\mathcal{P}$ over $\mathcal{H}_{\bar{m}}$ we count the hypotheses in $\mathcal{H}_{\bar{m}}$. There are $\binom{n_i-1}{m_i-1} \leq n_i^{m_i-1}$ possibilities to choose a cluster cardinality profile along a dimension i. (This is because each of the m_i clusters has a size of at least one. To define a cardinality profile we are free to distribute the "excess mass" of $n_i - m_i$ among the m_i clusters. The number of possible distributions equals the number of possibilities to place $m_i - 1$ ones in a sequence of $(n_i - m_i) + (m_i - 1) = n_i - 1$ ones and zeros.) For a fixed cardinality profile $hist(h|_i)$ there are $\binom{n_i}{|c_{i1}|,..,|c_{im_i}|} \leq e^{n_i H(hist(h|_i))}$ possibilities to assign x_i-s to the clusters. Putting the combinatorial calculations together we can define a distribution $\mathcal{P}$ over $\mathcal{H}_{\bar{m}}$ that satisfies (19).

To prove (20) we further define a uniform prior over $\mathcal{H}|_{y|\bar{m}}$. Note that there are $|\mathcal{Y}|^M$ possibilities to assign labels to the partition cells in $\mathcal{H}_{\bar{m}}$. Finally, we define a uniform prior over the choice of $\bar{m}$. There are n_i possibilities to chose the value of m_i (we can assign all x_i-s to a single cluster, assign each x_i to a separate cluster, and all the possibilities in between). Combining this with the combinatorial calculations performed for (19) yields (20). ■

Proof of Lemma 9 We first handle bound (21). We use the decomposition $KL(\mathcal{Q}\|\mathcal{P}) = -\mathbb{E}_{\mathcal{Q}}\ln\mathcal{P}(h) - H(\mathcal{Q})$ and bound $-\mathbb{E}_{\mathcal{Q}}\ln\mathcal{P}(h)$ and $H(\mathcal{Q})$ separately. We further decompose $\mathcal{P}(h) = \mathcal{P}(h|_1)\cdot..\cdot\mathcal{P}(h|_d)$ and $\mathcal{Q}(h)$ in a similar manner. Then $-\mathbb{E}_{\mathcal{Q}}\ln\mathcal{P}(h) = -\sum_i \mathbb{E}_Q \ln\mathcal{P}(h|_i)$ and $H(\mathcal{Q}) = \sum_i H(\mathcal{Q}(h|_i))$. Therefore, we can treat each dimension separately.

By Lemma 8:

$$-\mathbb{E}_{\mathcal{Q}}\ln\mathcal{P}(h|_i) \leq (m_i - 1)\ln n_i + n_i\mathbb{E}_{\mathcal{Q}}H(hist(h|_i)). \tag{23}$$

Hence, in order to bound $-\mathbb{E}_{\mathcal{Q}}\ln\mathcal{P}(h|_i)$ we have to bound the expected entropy of cluster cardinality profiles of the hypotheses generated by $\mathcal{Q}$. Recall that $\mathcal{Q}$ draws a cluster C_i for each $x_i \in \mathcal{X}_i$ according to $q(c_i|x_i)$ and that this process results in marginal distribution $\bar{q}(c_i) = \frac{1}{n_i}\sum_{x_i} q(c_i|x_i)$ over the normalized cluster sizes (this is where the uniform distribution over $\mathcal{X}_i$ comes in). To bound $\mathbb{E}_{\mathcal{Q}}H(hist(h|_i))$ we use the result on negative bias of empirical entropy estimates cited below, see Paninski (2003) for a proof.

Theorem 10 (Paninski, 2003) *Let $X_1,..,X_N$ be i.i.d. distributed by $p(x)$ and let $\hat{p}(x)$ be their empirical distribution. Then:*

$$\mathbb{E}_p H(\hat{p}) = H(p) - \mathbb{E}_p KL(\hat{p}\|p) \leq H(p). \tag{24}$$

By (24) $\mathbb{E}_{\mathcal{Q}}H(hist(h|_i)) \leq H(\bar{q}(c_i))$. Substituting this into (23) yields:

$$-\mathbb{E}_{\mathcal{Q}}\ln\mathcal{P}(h|_i) \leq n_i H(\bar{q}(c_i)) + (m_i - 1)\ln n_i. \tag{25}$$

Now we turn to bound $-H(\mathcal{Q}(h|_i)) = \mathbb{E}_{\mathcal{Q}}\ln\mathcal{Q}(h|_i)$. To do so we bound $\ln\mathcal{Q}(h|_i)$ from above. The bound follows from the fact that if we draw n_i values of C_i according to $q(c_i|x_i)$ the probability

of the resulting type is bounded from above by $e^{-n_i \bar{H}(C_i|X_i)}$, where $\bar{H}(C_i|X_i) = -\frac{1}{n_i}\sum_{x_i,c_i} q(c_i|x_i)\ln q(c_i|x_i)$ (see Cover and Thomas, 1991, Theorem 12.1.2). Thus, $\mathbb{E}_{\mathcal{Q}} \ln \mathcal{Q}(h|_i) \leq -n_i \bar{H}(C_i|X_i)$, which together with (25) and the identity $\bar{I}(X_i;C_i) = H(\bar{q}(c_i)) - \bar{H}(C_i|X_i)$ completes the proof of (21).

To prove (22) we recall that $\mathcal{Q}$ is defined for a fixed $\bar{m}$. Hence, $-\mathbb{E}_{\mathcal{Q}} \ln \mathcal{P}(h|_{y|\bar{m}}) = M \ln |\mathcal{Y}|$ and $-H(\mathcal{Q}(h|_{y|\bar{m}})) \leq 0$. Finally, since the prior $\mathcal{P}(\bar{m})$ over the selection of $\bar{m}$ is uniform we have $-\mathbb{E}_{\mathcal{Q}} \ln \mathcal{P}(\bar{m}) = \sum_{i=1}^{d} \ln n_i$ and $H(\mathcal{Q}(\bar{m})) = 0$, which is added to (21) by the additivity of $KL(\mathcal{Q}\|\mathcal{P})$ completing the proof. ■

3.4 PAC-Bayesian Analysis of Density Estimation with Grid Clustering

In this subsection we derive a generalization bound for density estimation with grid clustering. This time we have no labels and the goal is to find a good estimator for an unknown joint probability distribution $p(x_1,..,x_d)$ over a d-dimensional product space $\mathcal{X}_1 \times .. \times \mathcal{X}_d$ based on a sample of size N from p. As an illustrative example, think of estimating a joint probability distribution of words and documents (X_1 and X_2) from their co-occurrence matrix. The goodness of an estimator q for p is measured by $-\mathbb{E}_{p(x_1,..,x_d)} \ln q(X_1,..,X_d)$.

By Theorem 5, to obtain a meaningful bound for a direct estimation of $p(x_1,..,x_d)$ from $\hat{p}(x_1,..,x_d)$ we need N to be exponential in n_i-s, since the cardinality of the random variable $\langle X_1,..,X_d\rangle$ is $\prod_i n_i$. To reduce this dependency to be linear in $\sum_i n_i$ we restrict the estimator $q(X_1,..,X_d)$ to be of the factor form:

$$q(x_1,..,x_d) = \sum_{c_1,..,c_d} q(c_1,..,c_d) \prod_{i=1}^{d} q(x_i|c_i) \tag{26}$$

$$= \sum_{c_1,..,c_d} q(c_1,..,c_d) \prod_{i=1}^{d} \frac{q(x_i)}{q(c_i)} q(c_i|x_i). \tag{27}$$

We emphasize that the above decomposition assumption is only on the estimator q and not on the generating distribution p. A graphical model corresponding to Equation (26) is given in Figure 1.b. Similar to the model for discriminative prediction, this is also a two-level randomized prediction model.

We select the hypothesis space $\mathcal{H}$ to be the space of hard partitions of the product space $\mathcal{X}_1 \times .. \times \mathcal{X}_d$, as before; however, this time there are no labels to the partition cells. The general message of the following two theorems is that the empirical distribution over the coarse partition space $C_1 \times .. \times C_d$ converges to the true one faster than the empirical distribution over $\mathcal{X}_1 \times .. \times \mathcal{X}_d$ converges to its true counterpart. We also show that (27) can be used to extrapolate the distribution over cluster space back to $\mathcal{X}_1 \times .. \times \mathcal{X}_d$ space and obtain better generalization guarantees. Next we state this more formally.

As seen in the previous subsection, a distribution $\mathcal{Q} = \{q(c_i|x_i)\}_{i=1}^{d}$ is a distribution over $\mathcal{H}_{\bar{m}}$. To obtain a hypothesis $h \in \mathcal{H}_{\bar{m}}$ we draw a cluster for each $x_i \in \mathcal{X}_i$ according to $q(c_i|x_i)$. The way we have written (27) enables us to view it as a randomized prediction process: we draw a hypothesis h according to $\mathcal{Q}$ and then predict the probability of $\langle x_1,..,x_d\rangle$ as $q(c_1^h(x_1),..,c_d^h(x_d)) \prod_i \frac{q(x_i)}{q(c_i^h(x_i))}$, where $c_i^h(x_i) = h(x_i)$ is the partition cell that x_i fell within h. Although (27) skips the process of

drawing the complete partition h and returns the probability of $\langle x_1,..,x_d\rangle$ directly, the described randomized prediction process matches the predictions by (27) and thus enables its analysis with PAC-Bayesian bounds.

Let $h \in \mathcal{H}$ be a hard partition of $\mathcal{X}_1 \times .. \times \mathcal{X}_d$ and let $h(x_i)$ denote the cluster to which x_i is mapped in h. We define the distribution over the partition cells $\langle c_1,..,c_d\rangle$ induced by p and h:

$$p_h(c_1,..,c_d) = \sum_{\substack{x_1,..,x_d:\\ \forall i \;\; h(x_i)=c_i}} p(x_1,..,x_d),$$
$$p_h(c_i) = \sum_{x_i:h(x_i)=c_i} p(x_i).$$

We further define the distribution over partition cells induced by the empirical distribution $\hat{p}(x_1,..,x_d)$ corresponding to the sample and h by substitution of $\hat{p}$ instead of p in the above definitions:

$$\hat{p}_h(c_1,..,c_d) = \sum_{\substack{x_1,..,x_d:\\ \forall i \;\; h(x_i)=c_i}} \hat{p}(x_1,..,x_d),$$
$$\hat{p}_h(c_i) = \sum_{x_i:h(x_i)=c_i} \hat{p}(x_i).$$

We also define the distribution over partition cells induced by $\mathcal{Q}$ and p and its empirical counterpart:

$$p_{\mathcal{Q}}(c_1,..,c_d) = \sum_h \mathcal{Q}(h) p_h(c_1,..,c_d) = \sum_{x_1,..,x_d} p(x_1,..,x_d) \prod_{i=1}^d q(c_i|x_i),$$
$$p_{\mathcal{Q}}(c_i) = \sum_h \mathcal{Q}(h) p_h(c_i) = \sum_{x_i} p(x_i) q(c_i|x_i),$$

$$\hat{p}_{\mathcal{Q}}(c_1,..,c_d) = \sum_h \mathcal{Q}(h) \hat{p}_h(c_1,..,c_d) = \sum_{x_1,..,x_d} \hat{p}(x_1,..,x_d) \prod_{i=1}^d q(c_i|x_i),$$
$$\hat{p}_{\mathcal{Q}}(c_i) = \sum_h \mathcal{Q}(h) \hat{p}_h(c_i) = \sum_{x_i} \hat{p}(x_i) q(c_i|x_i).$$

We extrapolate $p_h, p_{\mathcal{Q}}, \hat{p}_h$ and $\hat{p}_{\mathcal{Q}}$ to the whole space $\mathcal{X}_1 \times .. \times \mathcal{X}_d$ using (27):

$$p_h(x_1,..,x_d) = p_h(c_1^h(x_1),..,c_d^h(x_d)) \prod_{i=1}^d \frac{p(x_i)}{p_h(c_i^h(x_i))},$$

$$p_{\mathcal{Q}}(x_1,..,x_d) = \sum_{c_1,..,c_d} p_{\mathcal{Q}}(c_1,..,c_d) \prod_{i=1}^d \frac{p(x_i)}{p_{\mathcal{Q}}(c_i)} q(c_i|x_i),$$

$$\hat{p}_h(x_1,..,x_d) = \hat{p}_h(c_1^h(x_1),..,c_d^h(x_d)) \prod_{i=1}^d \frac{\hat{p}(x_i)}{\hat{p}_h(c_i^h(x_i))},$$

$$\hat{p}_{\mathcal{Q}}(x_1,..,x_d) = \sum_{c_1,..,c_d} \hat{p}_{\mathcal{Q}}(c_1,..,c_d) \prod_{i=1}^d \frac{\hat{p}(x_i)}{\hat{p}_{\mathcal{Q}}(c_i)} q(c_i|x_i).$$

Note that $p_{\mathcal{Q}}(x_1,..,x_d)$ is a distribution over $\mathcal{X}_1 \times .. \times \mathcal{X}_d$, which has the form (27) and is the closest to the true distribution $p(x_1,..,x_d)$ under the constraint that $\{q(c_i|x_i)\}_{i=1}^d$ are fixed. Further, note

that since we have no access to $p(x_1,..,x_d)$ we do not know $p_{\mathcal{Q}}(x_1,..,x_d)$. In the next theorem we provide rates of convergence of the distributions $\hat{p}_{\mathcal{Q}}(x_1,..,x_d)$, $\hat{p}_{\mathcal{Q}}(c_1,..,c_d)$, and $\hat{p}_{\mathcal{Q}}(x_i)$ based on the sample to their counterparts corresponding to the true distribution $p(x_1,..,x_d)$.

Theorem 11 *For any probability measure p over $\mathcal{X}_1 \times .. \times \mathcal{X}_d$ and an i.i.d. sample S of size N according to p, with probability of at least $1-\delta$ for all grid clusterings $\mathcal{Q} = \{q(c_i|x_i)\}_{i=1}^d$ the following holds simultaneously*:

$$KL(\hat{p}_{\mathcal{Q}}(c_1,..,c_d)\|p_{\mathcal{Q}}(c_1,..,c_d)) \leq \frac{\sum_{i=1}^d n_i \bar{I}(X_i;C_i) + K_1}{N} \tag{28}$$

and for all i

$$KL(\hat{p}(x_i)\|p(x_i)) \leq \frac{(n_i-1)\ln(N+1) + \ln\frac{d+1}{\delta}}{N}, \tag{29}$$

where

$$K_1 = \sum_{i=1}^d m_i \ln n_i + (M-1)\ln(N+1) + \ln\frac{d+1}{\delta}. \tag{30}$$

As well, with probability greater than $1-\delta$:

$$KL(\hat{p}_{\mathcal{Q}}(x_1,..,x_d)\|p_{\mathcal{Q}}(x_1,..,x_d)) \leq \frac{\sum_{i=1}^d n_i \bar{I}(X_i;C_i) + K_2}{N}, \tag{31}$$

where

$$K_2 = \sum_{i=1}^d m_i \ln n_i + \left[M + \sum_{i=1}^d n_i - d - 1\right]\ln(N+1) - \ln\delta.$$

Before we discuss and prove the theorem we point out that although $\hat{p}_{\mathcal{Q}}(x_1,..,x_d)$ converges to $p_{\mathcal{Q}}(x_1,..,x_d)$ it still cannot be used to minimize $-\mathbb{E}_{p(x_1,..,x_d)}\ln\hat{p}_{\mathcal{Q}}(X_1,..,X_d)$, because it is not bounded from zero. Also, we cannot construct a density estimator by smoothing $\hat{p}_{\mathcal{Q}}(x_1,..,x_d)$ directly using Theorem 6, because the cardinality of the random variable $\langle X_1,..,X_d\rangle$ is $\prod_i n_i$ and this term will enter into the bounds. To circumvent this we take advantage of the factor form of $p_{\mathcal{Q}}$ and use the bounds (28) and (29). We define an estimator $\tilde{p}_{\mathcal{Q}}$, which is a smoothed version of $\hat{p}_{\mathcal{Q}}$ in the following way:

$$\begin{aligned}
\tilde{p}_h(c_1,..,c_d) &= \frac{\hat{p}_h(c_1,..,c_d)+\gamma}{1+\gamma M},\\
\tilde{p}(x_i) &= \frac{\hat{p}(x_i)+\gamma_i}{1+\gamma_i n_i},\\
\tilde{p}_h(c_i) &= \sum_{x_i:h(x_i)=c_i} \tilde{p}(x_i),\\
\tilde{p}_h(x_1,..,x_d) &= \tilde{p}_h(c_1^h(x_1),..,c_d^h(x_d)) \prod_{i=1}^d \frac{\tilde{p}(x_i)}{\tilde{p}_h(c_i^h(x_i))}.
\end{aligned} \tag{32}$$

And for a distribution $\mathcal{Q}$ over $\mathcal{H}$:

$$\tilde{p}_{\mathcal{Q}}(c_1,..,c_d) = \frac{\hat{p}_{\mathcal{Q}}(c_1,..,c_d)+\gamma}{1+\gamma M}, \tag{33}$$

$$\tilde{p}_{\mathcal{Q}}(c_i) = \sum_{x_i}\tilde{p}(x_i)q(c_i|x_i) = \frac{\hat{p}_{\mathcal{Q}}(c_i)+\gamma_i\bar{q}(c_i)n_i}{1+\gamma_i n_i}, \tag{34}$$

$$\begin{aligned}\tilde{p}_{\mathcal{Q}}(x_1,..,x_d) &= \sum_h \mathcal{Q}(h)\tilde{p}_h(x_1,..,x_d)\\ &= \sum_{c_1,..,c_d}\tilde{p}_{\mathcal{Q}}(c_1,..,c_d)\prod_{i=1}^{d}\frac{\tilde{p}(x_i)}{\tilde{p}_{\mathcal{Q}}(c_i)}q(c_i|x_i).\end{aligned} \tag{35}$$

In the following theorem we provide a bound on $-\mathbb{E}_{p(x_1,..,x_d)}\ln\tilde{p}_{\mathcal{Q}}(X_1,..,X_d)$. Note that we take the expectation with respect to the true, unknown distribution p that may have an arbitrary form (i.e., p is not restricted to be of the factor form (26)).

Theorem 12 *For the density estimator $\tilde{p}_{\mathcal{Q}}(x_1,..,x_d)$ defined by equations* (32), (33), (34), *and* (35), *$-\mathbb{E}_{p(x_1,..,x_d)}\tilde{p}_{\mathcal{Q}}(X_1,..,X_d)$ attains its minimum at $\gamma(\mathcal{Q}) = \frac{\sqrt{\varepsilon(\mathcal{Q})/2}}{M}$ and $\gamma_i = \frac{\sqrt{\varepsilon_i/2}}{n_i}$, where $\varepsilon(\mathcal{Q})$ is defined by the right-hand side of* (28) *and ε_i-s are defined by the right-hand side of* (29). *At this optimal level of smoothing, with probability greater than $1-\delta$ for all $\mathcal{Q} = \{q(c_i|x_i)\}_{i=1}^d$ simultaneously*:

$$-\mathbb{E}_{p(x_1,..,x_d)}\ln\tilde{p}_{\mathcal{Q}}(X_1,..,X_d) \le -I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d)) + \ln(M)\sqrt{\frac{\sum_{i=1}^d n_i\bar{I}(X_i;C_i)+K_1}{2N}} + K_3, \tag{36}$$

where $I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d)) = \left[\sum_{i=1}^d H(\hat{p}_{\mathcal{Q}}(c_i))\right] - H(\hat{p}_{\mathcal{Q}}(c_1,..,c_d))$ is the multi-information between $C_1,..,C_d$ with respect to $\hat{p}_{\mathcal{Q}}(c_1,..,c_d)$, K_1 is defined by (30),

$$K_3 = \phi(\varepsilon(\mathcal{Q})) + \left[\sum_{i=1}^d H(\hat{p}(x_i)) + 2\sqrt{\varepsilon_i/2}\ln n_i + \phi(\varepsilon_i) + \psi(\varepsilon_i)\right],$$

and the functions ϕ and ψ are defined in Theorem 6.

Discussion: We discuss Theorem 12 first. We point out that $\tilde{p}_{\mathcal{Q}}(x_1,..,x_d)$ is directly related to $\hat{p}_{\mathcal{Q}}(x_1,..,x_d)$ and that $\hat{p}_{\mathcal{Q}}(x_1,..,x_d)$ is determined by the empirical frequencies $\hat{p}(x_1,..,x_d)$ of the sample and our choice of $\mathcal{Q} = \{q(c_i|x_i)\}_{i=1}^d$. There are only two quantities in the bound (36) that depend on the choice of $\mathcal{Q}$: $-I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d))$ and $\sum_i \frac{n_i}{N}\bar{I}(X_i;C_i)$ [note that the latter also appears in $\phi(\varepsilon(\mathcal{Q})$ in K_3]. Thus, Theorem 12 suggests that a good estimator $\tilde{p}_{\mathcal{Q}}(x_1,..,x_d)$ of $p(x_1,..,x_d)$ should optimize the trade-off between $-I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d))$ and $\sum_i \frac{n_i}{N}\bar{I}(X_i;C_i)$. Similar to Theorem 7, the latter term corresponds to the mutual information that the hidden cluster variables preserve on the observed variables. Larger values of $\bar{I}(X_i;C_i)$ correspond to partitions of $\mathcal{X}_1,..,\mathcal{X}_d$, which are more complex. The first term, $-I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d))$, corresponds to the amount of structural information on C_i-s extracted by the partition. More precisely, we need to look at the value of $\sum_i H(\hat{p}(x_i)) - I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d))$, where $\sum_i H(\hat{p}(x_i))$ is a part of K_3 and roughly corresponds to the performance we can achieve by approximating $p(x_1,..,x_d)$ with a product of empirical marginals $\prod_i \hat{p}(x_i)$. Thus, $-I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d))$ is the added value of the partition in estimating $p(x_1,..,x_d)$ and since $\sum_i H(\hat{p}(x_i)) \ge I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d))$ the bound (36) is always positive.

The value of $I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d))$ increases monotonically with the increase of the partition complexity $\mathcal{Q}$ (we can see this by the information processing inequality, Cover and Thomas, 1991). Thus, the trade-off in (36) is analogous to the trade-off in (18): the partition $\mathcal{Q}$ should balance its utility function $-I(\hat{p}_{\mathcal{Q}}(c_1,..,c_d))$ and the statistical reliability of the estimate of the utility function, which is related to $\sum_i \frac{n_i}{N} \bar{I}(X_i;C_i)$. This trade-off suggests a modification to the original objective of co-clustering in Dhillon et al. (2003), which is maximization of $I(C_1;C_2)$ alone (Dhillon et al., 2003 discuss the case of two-dimensional matrices). The trade-off in (36) can be applied to model order selection.

Now we make a few comments concerning Theorem 11. An interesting point about this theorem is that the cardinality of the random variable $\langle X_1,..,X_d \rangle$ is $\prod_i n_i$. Thus, a direct application of Theorem 2 to bound $KL(\hat{p}_{\mathcal{Q}}(x_1,..,x_d)\|p_{\mathcal{Q}}(x_1,..,x_d))$ would introduce this term into the bound. However, by using the factor form (27) of $\hat{p}_{\mathcal{Q}}(x_1,..,x_d)$ and $p_{\mathcal{Q}}(x_1,..,x_d)$ we are able to reduce this dependency to $(M+\sum_i n_i - d - 1)$. This result reveals the great potential of applying PAC-Bayesian analysis to more complex graphical models, which we explore further in Section 7.

3.4.1 PROOFS

We conclude this section by presenting the proofs of Theorems 11 and 12.

Proof of Theorem 11 The proof is based on PAC-Bayesian theorem on density estimation (Theorem 2). To apply the theorem we need to define a prior $\mathcal{P}$ over $\mathcal{H}$ and then calculate $KL(\mathcal{Q}\|\mathcal{P})$. We note that for a fixed $\mathcal{Q}$ the cardinalities of the clusters $\bar{m}$ are fixed. There are $\prod_i n_i$ disjoint subspaces $\mathcal{H}_{\bar{m}}$ in $\mathcal{H}$. We handle each $\mathcal{H}_{\bar{m}}$ independently and then combine the results to obtain Theorem 11.

By Theorem 2 and Lemma 9, for the prior $\mathcal{P}$ over $\mathcal{H}_{\bar{m}}$ defined in Lemma 8, with probability greater than $1-\frac{\delta}{(d+1)\prod_i n_i}$ we obtain (28) for each $\mathcal{H}_{\bar{m}}$. In addition, by Theorem 5 with probability greater than $1-\frac{\delta}{d+1}$ inequality (29) holds for each X_i. By a union bound over the $\prod_i n_i$ subspaces of $\mathcal{H}$ and the d variables X_i we obtain that (28) and (29) hold simultaneously for all $\mathcal{Q}$ and X_i with probability greater than $1-\delta$.

To prove (31), fix some hard partition h and let $c_i^h = h(x_i)$. Then:

$$
\begin{aligned}
&KL(\hat{p}_h(x_1,..,x_d)\|p_h(x_1,..,x_d)) \\
&\quad = KL(\hat{p}_h(x_1,..,x_d,c_1^h(x_1),..,c_d^h(x_d))\|p_h(x_1,..,x_d,c_1^h(x_1),..,c_d^h(x_d))) \\
&\quad = KL(\hat{p}_h(c_1,..,c_d)\|p_h(c_1,..,c_d)) \\
&\qquad + KL(\hat{p}_h(x_1,..,x_d|c_1^h(x_1),..,c_d^h(x_d))\|p_h(x_1,..,x_d|c_1^h(x_1),..,c_d^h(x_d))) \\
&\quad = KL(\hat{p}_h(c_1,..,c_d)\|p_h(c_1,..,c_d)) + \sum_{i=1}^{d} KL(\hat{p}_h(x_i|c_i^h(x_i))\|p_h(x_i|c_i^h(x_i))) \\
&\quad = KL(\hat{p}_h(c_1,..,c_d)\|p_h(c_1,..,c_d)) + \sum_{i=1}^{d} KL(\hat{p}(x_i)\|p(x_i)) - \sum_{i=1}^{d} KL(\hat{p}_h(c_i)\|p_h(c_i)) \\
&\quad \leq KL(\hat{p}_h(c_1,..,c_d)\|p_h(c_1,..,c_d)) + \sum_{i=1}^{d} KL(\hat{p}(x_i)\|p(x_i)).
\end{aligned}
$$

And:

$$\begin{aligned}\mathbb{E}_S e^{NKL(\hat{p}_h(x_1,..,x_d)\|p_h(x_1,..,x_d))} &\leq \left(\mathbb{E}_S e^{NKL(\hat{p}_h(c_1,..,c_d)\|p_h(c_1,..,c_d))}\right)\prod_{i=1}^{d}\mathbb{E}_S e^{NKL(\hat{p}(x_i)\|p(x_i))}\\ &\leq (N+1)^{M+\sum_{i=1}^{d}n_i-(d+1)},\end{aligned}$$

where the last inequality is by Theorem 3. From here we follow the lines of the proof of Theorem 2. Namely:

$$\begin{aligned}\mathbb{E}_S\left[\mathbb{E}_{\mathcal{P}(h)}e^{NKL(\hat{p}_h(x_1,..,x_d)\|p_h(x_1,..,x_d))}\right] &= \mathbb{E}_{\mathcal{P}(h)}\left[\mathbb{E}_S e^{NKL(\hat{p}_h(x_1,..,x_d)\|p_h(x_1,..,x_d))}\right]\\ &\leq (N+1)^{M+\sum_{i=1}^{d}n_i-(d+1)}.\end{aligned}$$

Thus, by Markov's inequality $\mathbb{E}_{\mathcal{P}(h)}e^{NKL(\hat{p}_h(x_1,..,x_d)\|p_h(x_1,..,x_d))} \leq \frac{1}{\delta}(N+1)^{M+\sum_i n_i-(d+1)}$ with probability of at least $1-\delta$ and (31) follows by the change of measure inequality (8) and convexity of the KL-divergence, when the prior $\mathcal{P}$ over $\mathcal{H}$ defined in Lemma 8 is selected (this time we give a weight of $(\prod_i n_i)^{-1}$ to each $\mathcal{H}_{\bar{m}}$ and obtain a prior over the whole $\mathcal{H}$). The calculation of $KL(\mathcal{Q}\|\mathcal{P})$ for this prior is provided in Lemma 9. ■

Proof of Theorem 12

$$\begin{aligned}-\mathbb{E}_{p(x_1,..,x_d)}\ln\tilde{p}_{\mathcal{Q}}(X_1,..,X_d) &= -\mathbb{E}_{p(x_1,..,x_d)}\ln\mathbb{E}_{\mathcal{Q}(h)}\tilde{p}_h(X_1,..,X_d)\\ &\leq -\mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{p(x_1,..,x_d)}\ln\tilde{p}_h(X_1,..,X_d)\\ &= -\mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{p(x_1,..,x_d)}\ln\tilde{p}_h(C_1^h(X_1),..,C_d^h(X_d))\prod_i\frac{\tilde{p}(X_i)}{\tilde{p}_h(C_i^h(X_i))}\\ &= -\mathbb{E}_{\mathcal{Q}(h)}\left[\mathbb{E}_{p_h(c_1,..,c_d)}\ln\tilde{p}_h(C_1,..,C_d)\right]-\sum_i\mathbb{E}_{p(x_i)}\ln\tilde{p}(X_i)+\sum_i\mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{p_h(c_i)}\ln\tilde{p}_h(C_i)\\ &\leq -\mathbb{E}_{\mathcal{Q}(h)}\left[\mathbb{E}_{p_h(c_1,..,c_d)}\ln\tilde{p}_h(C_1,..,C_d)\right]-\sum_i\mathbb{E}_{p(x_i)}\ln\tilde{p}(X_i)+\sum_i\mathbb{E}_{p_{\mathcal{Q}}(c_i)}\ln\tilde{p}_{\mathcal{Q}}(C_i)\end{aligned}$$

At this point we use (14) to bound the first and the second term and the lower bound (16) to bound the last term and obtain (36). ■

4. Algorithms

In the previous section we presented generalization bounds for discriminative prediction and density estimation with co-clustering. The bounds presented in Theorems 7 and 12 hold for any prediction rule $\mathcal{Q}$ based on grid clustering of the parameter space $\mathcal{X}_1 \times .. \times \mathcal{X}_d$. In Seldin (2009) it is shown that for $d = 1$ the global minimum of bound (18) can be found efficiently. However, for $d \geq 2$ it can be exponentially hard to find the global minimum of both (18) and (36). Further, although we show in the applications section that bound (18) is remarkably tight, its tightness can still be insufficient for practical purposes. In this section we suggest how to replace the bounds with parameterized trade-offs that can be further fine-tuned, for example, via cross-validation, to improve the usability

in practice. The substitution of the bounds with parametrized trade-offs does not compromise on the rigor of the analysis: first, the bounds hold simultaneously for all the solutions found by minimization of the parameterized trade-offs and, second, the parameter of the trade-offs can also be tuned by substituting the result of trade-off minimization back into the corresponding bound, thus providing an estimate on a local minima of the bound.

4.1 Minimization of the PAC-Bayesian Bound for Discriminative Prediction with Grid Clustering

We start with minimization of the PAC-Bayesian bound for discriminative prediction based on grid clustering (18) suggested in Theorem 7. We rewrite the bound in a slightly different way in order to separate terms which are independent of the conditional distributions in $\mathcal{Q}$:

$$kl(\hat{L}(\mathcal{Q})\|L(\mathcal{Q})) \leq \frac{\sum_{i=1}^{d} n_i \bar{I}(X_i;C_i) + K}{N}, \tag{37}$$

where

$$K = \sum_{i=1}^{d} m_i \ln n_i + M \ln |\mathcal{Y}| + \frac{1}{2}\ln(4N) - \ln \delta. \tag{38}$$

Note that K depends on the number of clusters m_i used along each dimension, but not on a specific form of a grid partition. Once the number of clusters used along each dimension has been selected, K is constant.

The minimization problem corresponding to (37) can be stated as follows:

$$\min_{\mathcal{Q}} \; L \quad s.t. \quad kl(\hat{L}(\mathcal{Q})\|L) = \frac{\sum_{i=1}^{d} n_i \bar{I}(X_i;C_i) + K}{N}. \tag{39}$$

It is generally possible to find a local minimum of the minimization problem (39) directly using alternating projection methods - see, for example, Germain et al. (2009) for such an approach to solving a similar minimization problem for linear classifiers. We take a slightly different approach that further enables us to compensate for the imperfection of the bounds. Since K is constant, $L(\mathcal{Q})$ depends on a parameterized trade-off between $\hat{L}(\mathcal{Q})$ and $\sum_{i=1}^{d} n_i \bar{I}(X_i;C_i)$, which can be written as follows:

$$\mathcal{F}(\mathcal{Q},\beta) = \beta N \hat{L}(\mathcal{Q}) + \sum_{i=1}^{d} n_i \bar{I}(X_i;C_i), \tag{40}$$

where β is the trade-off parameter. The corresponding minimization problem is:

$$\mathcal{F}_{min}(\beta) = \min_{\mathcal{Q}} \beta N \hat{L}(\mathcal{Q}) + \sum_{i=1}^{d} n_i \bar{I}(X_i;C_i). \tag{41}$$

In general, every value of β yields a different solution to the minimization problem (41). The optimum of (39) (which is computationally hard to find) corresponds to some specific value of β. Hence, by scanning the possible values of β, minimizing (41), and substituting the resulting $\hat{L}(\mathcal{Q})$ and $\bar{I}(X_i;C_i)$ back into (18) it is virtually possible to find the optimum of (39) (only virtually, because finding the global optimum of (41) is computationally hard as well). However, the trade-off (40) provides an additional degree of freedom. In cases where the bound (18) is not sufficiently

tight for practical applications it is possible to tune the trade-off by determining the desired value of β via cross-validation instead of back-substitution into the bound.

The minimization problem (41) is closely related to the rate distortion trade-off in information theory (Cover and Thomas, 1991). To find a local minimum of $\mathcal{F}(\mathcal{Q},\beta)$ we adopt an EM-like alternating projection procedure, very similar to the Blahut-Arimoto algorithm for minimization of the rate distortion function (Arimoto, 1972; Blahut, 1972; Cover and Thomas, 1991). We note that for $d \geq 2$ the alternating projections involve more than two convex sets and hence only a local minimum can be achieved. (For $d = 1$ the procedure achieves the global minimum.) For the sake of simplicity of the notations we restrict ourselves to the case of $d = 2$, but it is straightforward to extend the algorithm to higher dimensions.

The Lagrangian corresponding to the minimization problem (41) is:

$$\mathcal{L}(\mathcal{Q},\beta) = \beta N \hat{L}(\mathcal{Q}) + \sum_{i=1}^{2} n_i \bar{I}(X_i;C_i) + \sum_{i=1}^{2} \sum_{x_i \in \mathcal{X}_i} \nu(x_i) \sum_{c_i} q(c_i|x_i) + \sum_{c_1,c_2} \nu(c_1,c_2) \sum_{y} q(y|c_1,c_2),$$

where ν-s are Lagrange multipliers corresponding to normalization constraints on $\{q(c_i|x_i)\}_{i=1}^{2}$ and $q(y|c_1,c_2)$. In order to minimize $\mathcal{L}(\mathcal{Q},\beta)$ we write $\hat{L}(\mathcal{Q})$ explicitly:

$$\begin{aligned}
\hat{L}(\mathcal{Q}) &= \sum_{x_1,x_2,y} \hat{p}(x_1,x_2,y) \sum_{y'} q(y'|x_1,x_2) l(y,y') \\
&= \sum_{x_1,x_2,y} \hat{p}(x_1,x_2,y) \sum_{y',c_1,c_2} q(y'|c_1,c_2) q(c_1|x_1) q(c_2|x_2) l(y,y') \\
&= \sum_{y,y'} l(y,y') \sum_{c_1,c_2} q(y'|c_1,c_2) \sum_{x_1,x_2} q(c_1|x_1) \hat{p}(x_1,x_2,y) q(c_2|x_2).
\end{aligned}$$

We further derive $\hat{L}(\mathcal{Q})$ with respect to $q(c_1|x_1)$. The derivative with respect to $q(c_2|x_2)$ is similar.

$$\frac{\partial \hat{L}(\mathcal{Q})}{\partial q(c_1|x_1)} = \sum_{y,y'} l(y,y') \sum_{x_2,c_2} q(y'|c_1,c_2) \hat{p}(x_1,x_2,y) q(c_2|x_2). \tag{42}$$

Recall that $\bar{I}(X_i;C_i) = \frac{1}{n_i} \sum_{x_i,c_i} q(c_i|x_i) \ln \frac{q(c_i|x_i)}{\bar{q}(c_i)}$ and $\bar{q}(c_i) = \frac{1}{n_i} \sum_{x_i} q(c_i|x_i)$. Hence the derivative of $n_i \bar{I}(X_i;C_i)$ is:

$$\frac{\partial n_i \bar{I}(X_i;C_i)}{\partial q(c_i|x_i)} = \ln \frac{q(c_i|x_i)}{\bar{q}(c_i)}.$$

Derivatives of the remaining terms in $\mathcal{L}(\mathcal{Q},\beta)$ provide normalization for the corresponding variables. Thus, by taking the derivative of $\mathcal{L}(\mathcal{Q},\beta)$ with respect to $q(c_i|x_i)$, equating it to zero and reorganizing the terms we obtain a set of self-consistent equations that can be iterated until conver-

gence:

$$\bar{q}_t(c_i) = \frac{1}{n_i}\sum_{x_i} q_t(c_i|x_i),$$
$$q_{t+1}(c_i|x_i) = \frac{\bar{q}_t(c_i)}{Z_{t+1}(x_i)} e^{-\beta N \frac{\partial \hat{L}(\mathcal{Q}_t)}{\partial q(c_i|x_i)}},$$
$$Z_{t+1}(x_i) = \sum_{c_i} q_{t+1}(c_i|x_i),$$
$$y^*_{t+1}(c_1,c_2) = \arg\min_{y'} \sum_{y} l(y,y') \sum_{x_1,x_2} q_{t+1}(c_1|x_1)\hat{p}(x_1,x_2,y)q_{t+1}(c_2|x_2), \tag{43}$$
$$q_{t+1}(y|c_1,c_2) = \delta[y, y^*_{t+1}(c_1,c_2)], \tag{44}$$

where $\delta[\cdot,\cdot]$ is the Kronecker delta function, $\frac{\partial \hat{L}(\mathcal{Q})}{\partial q(c_i|x_i)}$ is given by (42), and the subindex t denotes the iteration number. Equations (43) and (44) correspond to minimization of $\hat{L}(\mathcal{Q})$ with respect to $q(y|c_1,c_2)$ and generally depend on the loss function. For the zero-one loss, $y^*(c_1,c_2)$ is the most frequent value of y appearing in the $\langle c_1,c_2\rangle$ partition cell; for the absolute loss it is the median value; for the quadratic loss it is the average value. We summarize the algorithm in the Algorithm 1 box.[2] We note that for the quadratic loss the loss minimizer $y^*(c_1,c_2)$, which is the average value in this case, can fall out of the finite space of labels $\mathcal{Y}$. However, the algorithm can still be applied and a bound can be obtained by post-process quantization, see appendix B for details.

Algorithm 1 Algorithm for minimization of $\mathcal{F}(\mathcal{Q},\beta) = \beta N \hat{L}(\mathcal{Q}) + \sum_{i=1}^{2} n_i \bar{I}(X_i;C_i)$ by alternating projections.

Input: $\hat{p}(x_1,x_2,y), N, n_1, n_2, m_1, m_2, l(y,y'), |\mathcal{Y}|, \beta$.
Initialize: $q_0(c_i|x_i)$ and $q_0(y|c_1,c_2)$ randomly.
$t \leftarrow 0$
$\bar{q}_t(c_i) \leftarrow \frac{1}{n_i}\sum_{x_i} q_t(c_i|x_i)$
repeat
  for $i = 1,2$ **do**
    $q_{t+1}(c_i|x_i) \leftarrow \bar{q}_t(c_i) e^{-\beta N \frac{\partial \hat{L}(\mathcal{Q}_t)}{\partial q(c_i|x_i)}}$
    $Z_{t+1}(x_i) \leftarrow \sum_{c_i} q_{t+1}(c_i|x_i)$
    $q_{t+1}(c_i|x_i) \leftarrow \frac{q_{t+1}(c_i|x_i)}{Z_{t+1}(x_i)}$
    $\bar{q}_{t+1}(c_i) \leftarrow \frac{1}{n_i}\sum_{x_i} q_{t+1}(c_i|x_i)$
    $y^*_{t+1}(c_1,c_2) \leftarrow \arg\min_{y'} \sum_{y} l(y,y') \sum_{x_1,x_2} q_{t+1}(c_1|x_1)\hat{p}(x_1,x_2,y)q_{t+1}(c_2|x_2)$
    $q_{t+1}(y|c_1,c_2) \leftarrow \delta[y, y^*_{t+1}(c_1,c_2)]$
    $t \leftarrow t+1$
  end for
until convergence
return $\{q_t(c_i|x_i)\}_{i=1}^2, q_t(y|c_1,c_2)$ from the last iteration.

2. Matlab implementation of the algorithm is available at `http://www.kyb.mpg.de/~seldin`.

4.2 Minimization of the PAC-Bayesian Bound for Density Estimation

Similar to the PAC-Bayesian bound for discriminative prediction, the PAC-Bayesian bound for density estimation (36) depends on the trade-off:

$$\mathcal{G}(\mathcal{Q},\beta) = -\beta N I(\hat{p}_{\mathcal{Q}}(c_1,c_2)) + \sum_{i=1}^{2} n_i \bar{I}(X_i;C_i).$$

All other terms in (36) do not depend on the specific form of grid partition $\mathcal{Q}$. (As in the previous subsection we restrict ourselves to $d=2$.) Unfortunately, $-I(\hat{p}_{\mathcal{Q}}(c_1,c_2))$ is concave in $q(c_i|x_i)$-s, whereas $\bar{I}(X_i;C_i)$ is convex in $q(c_i|x_i)$. Therefore, alternating projection methods are hard to apply. Instead, $\mathcal{G}(\mathcal{Q},\beta)$ can be minimized (with respect to $\mathcal{Q}$) using sequential minimization (Slonim et al., 2002; Dhillon et al., 2003). The essence of the sequential minimization method is that we start with some random assignment $q(c_i|x_i)$ and then iteratively take x_i-s out of their clusters and reassign them to new clusters, so that $\mathcal{G}(\mathcal{Q},\beta)$ is minimized. This approach leads to a hard partition of the data (i.e., each x_i is deterministically assigned to a single c_i). The algorithm is given in Algorithm 2 box.

Algorithm 2 Algorithm for sequential minimization of $\mathcal{G}(\mathcal{Q},\beta) = -\beta N I(\hat{p}_{\mathcal{Q}}(c_1,c_2)) + \sum_{i=1}^{2} n_i \bar{I}(X_i;C_i)$.

Input: $\hat{p}(x_1,x_2), N, n_1, n_2, m_1, m_2, \beta$.
Initialize: $q_0(c_i|x_i)$ randomly.
repeat
  for all $x_1 \in \mathcal{X}_1$ and all $x_2 \in \mathcal{X}_2$ according to some random order over $\mathcal{X}_1$ and $\mathcal{X}_2$ **do**
    for $i = 1,2$ **do**
      Select $x_i \in \mathcal{X}_i$ according to the order selected above.
      Compute $\mathcal{G}(\mathcal{Q},\beta)$ for each possible assignment of x_i to $c_i \in \{1,..,m_i\}$
      Reassign x_i to c_i such that $\mathcal{G}(\mathcal{Q},\beta)$ is minimized.
      Update $\hat{p}_{\mathcal{Q}}(c_1,c_2) \leftarrow \sum_{x_1,x_2} q(c_1|x_1)\hat{p}(x_1,x_2)q(c_2|x_2)$.
    end for
  end for
until no reassignments further minimize $\mathcal{G}(\mathcal{Q},\beta)$.
return $\{q(c_i|x_i)\}_{i=1}^{2}$ from the last iteration.

5. Applications

In this section we illustrate an application of the PAC-Bayesian bound for discriminative prediction based on co-clustering (18) and Algorithm 1 for minimization of the corresponding trade-off (40) on the problem of collaborative filtering. The problem of collaborative filtering was discussed in the previous sections. The goal of collaborative filtering is to complete the missing entries in a viewers-by-movies ratings matrix. This problem attracted a great deal of attention recently thanks to the Netflix challenge.[3] Since our goal here is mainly to illustrate our approach to co-clustering via the PAC-Bayesian bounds rather than to solve the large-scale challenge we concentrate on a

3. See `http://www.netflixprize.com/rules`.

much smaller MovieLens 100K data set.[4] The data set consists of 100,000 ratings on a five-star scale for 1,682 movies by 943 users. We take the five non-overlapping splits of the data set into 80% training and 20% test subsets provided at the MovieLens website. We stress that the training data are extremely sparse - only 5% of the training matrix entries are populated, whereas 95% of the values are missing.

To measure the accuracy of our algorithm we use the mean absolute error (MAE) measure, which is commonly used for evaluation on this data set (Herlocker et al., 2004). Let $\check{p}(x_1, x_2, y)$ be the distribution over $\langle X_1, X_2, Y \rangle$ in the test set. The mean absolute error is defined as:

$$MAE = \sum_{x_1, x_2, y} \check{p}(x_1, x_2, y) \sum_{y'} q(y'|x_1, x_2)|y - y'|.$$

In previous work the best MAE reported for this data set was 0.73 (Herlocker et al., 2004). It is worth recalling that the ratings are on a five-star scale, thus a MAE of 0.73 means that, on average, the predicted rating is 0.73 stars (less than one star) far from the observed rating. The maximal possible error is 4 (which occurs if we predict one star instead of five or vice versa), which determines the scale on which all the results should be judged.

In Seldin et al. (2007) we improved the MAE on this data set to 0.72 by using a Minimum Description Length (MDL, Grünwald, 2007) formulation of co-clustering. In the MDL formulation the co-clustering solutions are evaluated by the total description length, which includes the length of the description of assignments of x_i-s to c_i-s together with the length of the description of the ratings given the assignments. For fixed numbers of clusters (m_i-s) used along each dimension, the MDL solution corresponds to optimization of the trade-off (40) with logarithmic loss $\hat{L}(\mathcal{Q}) = \hat{I}(Y; C_1, C_2)$ and $\beta = 1$ (where $\hat{I}(Y; C_1, C_2)$ is the empirical mutual information between the clusters and the label). In the MDL formulation of co-clustering developed in Seldin et al. (2007) only hard (deterministic) assignments of x_i-s to c_i-s were considered. The best performance of 0.72 was achieved at $m_1 = 13$ and $m_2 = 6$ with below 1% sensitivity to small changes in m_1 and m_2 both in the description length and in the prediction accuracy. The deviation in prediction accuracy between the five splits of the MovieLens data set was below 0.01.

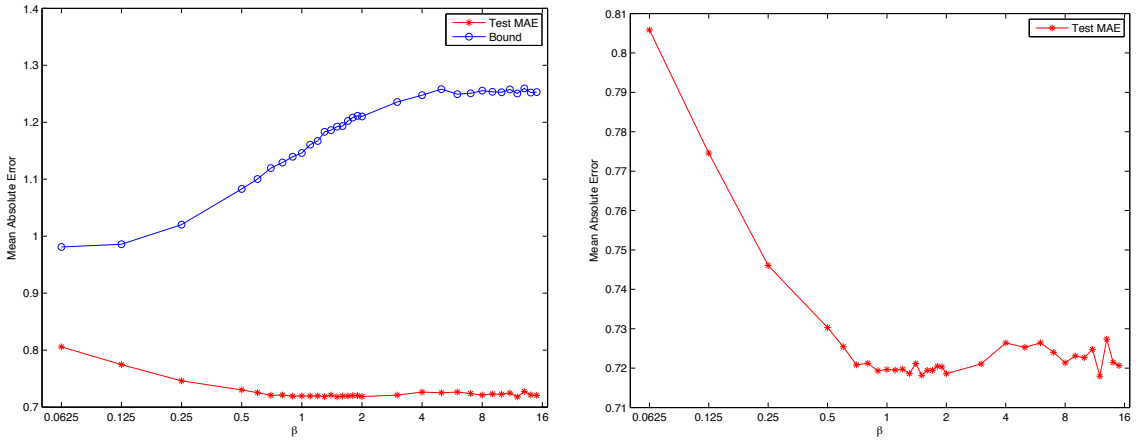
In the present work we implemented Algorithm 1 for minimization of $\mathcal{F}(\mathcal{Q}, \beta)$ as a function of $\mathcal{Q}$ and applied it to the MovieLens data set. There are four major differences between Algorithm 1 and the MDL algorithm suggested in Seldin et al. (2007) that should be highlighted:

- Algorithm 1 directly optimizes a given loss function (MAE in the case of MovieLens) rather than the description length, which is only indirectly related to the loss function.

- Algorithm 1 considers soft assignments of x_i-s to c_i-s.

- Algorithm 1 is an iterative projection algorithm rather than the sequential optimization algorithm suggested in Seldin et al. (2007). Note that this point is neither positive nor negative, since sequential optimization algorithms are very powerful and especially in hard cases can outperform iterative projection methods. The advantage of iterative projection methods is in their mathematical elegance, faster convergence (although in the hard cases it may be fast convergence to trivial, but strong attractors), and the ability to handle soft assignments.

4. Available at `http://www.grouplens.org`.



(a) Bound (18). (b) Test Loss (zoom into subfigure a.).

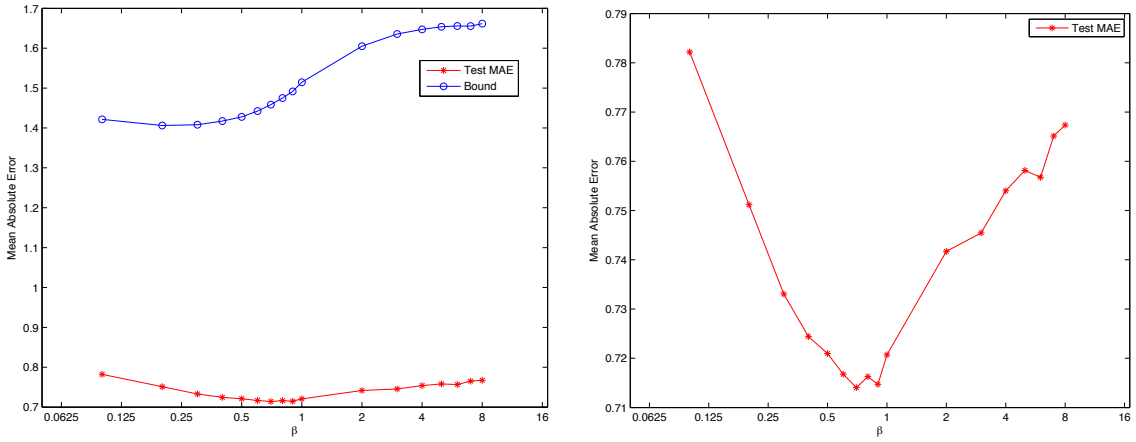
Figure 3: Co-clustering of the MovieLens data set into 13x6 clusters. Figure (a) shows the value of bound (18) together with the MAE on the test set as a function of β. Figure (b) zooms into MAE on the test set. The values of β are on a log scale. See text for further details.

- Algorithm 1 considers arbitrary values of β. (However, the algorithm in Seldin et al. (2007) can be easily extended to handle arbitrary values of β.) As we will show below, the value of $\beta = 1$ dictated by the MDL formulation is not always optimal and MDL solutions can overfit the data. This observation was already made previously in a context of other learning problems by Kearns et al. (1997).

We conducted three experiments with Algorithm 1. In all three experiments we fixed the numbers of clusters m_1 and m_2 used along both dimensions and analyzed the MAE on the test set and the value of bound (18) as a function of β. In each experiment, for each of the five splits of the data set into training and test sets mentioned earlier, and for each value of β we applied 10 random initializations of the algorithm. The solution $\mathcal{Q}$ corresponding to the best value of $\mathcal{F}(\mathcal{Q}, \beta)$ per each data split and per each value of β was then selected. We further calculated the average of the results over the data set splits to produce the graphs of the bound values and test MAE as functions of β.

In the first experiment we verified that we are able to reproduce the results achieved previously in Seldin et al. (2007). We set $m_1 = 13$ and $m_2 = 6$, as the best values obtained in Seldin et al. (2007) and applied Algorithm 1. The results are presented in Figure 3. We make the following conclusions from this experiment:

- The performance of Algorithm 1 is comparable to the performance achieved in Seldin et al. (2007) with sequential optimization.
- The optimal performance is achieved at β close to one, which corresponds to the MDL functional optimized in Seldin et al. (2007).
- The values of the bound are meaningful (recall that the maximal possible loss is 4; thus the bound value of ~ 1.25 is informative).
- The bound is 25%-75% far from the test error.



(a) Bound (18). (b) Test Loss (zoom into subfigure a.).

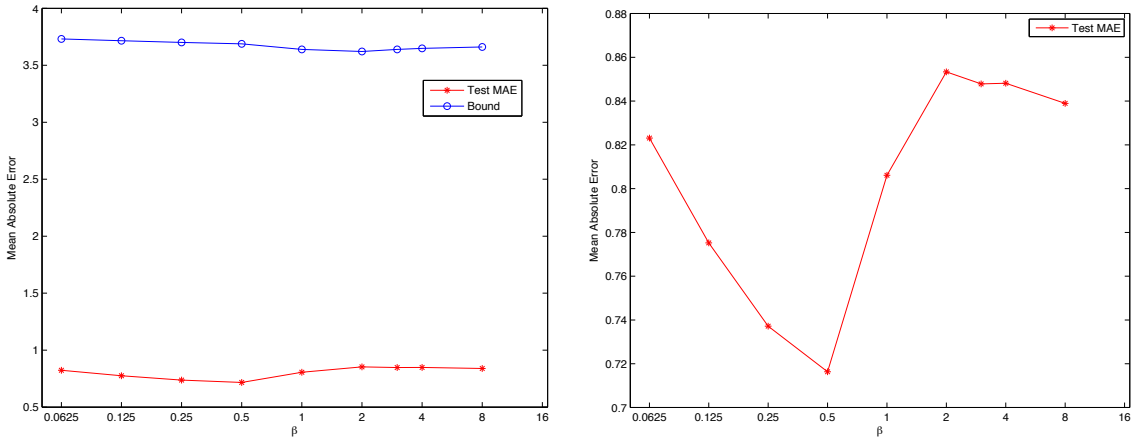
Figure 4: Co-clustering of the MovieLens data set into 50x50 clusters. Figure (a) shows the value of bound (18) together with the MAE on the test set as a function of β. Figure (b) zooms into MAE on the test set. The values of β are on a log scale. See text for further details.

- The bound does not follow the shape of the test loss. According to the bound in this task it is best to assign all the data to one big cluster. This is explained by the fact that this is a hard problem and the improvement in the empirical loss $\hat{L}(\mathcal{Q})$ achieved by co-clustering is relatively small. For the best co-clustering solution found $\hat{L}(\mathcal{Q}) \approx 0.67$, whereas if we assign all the data to one big cluster $\hat{L}(\mathcal{Q}) \approx 0.89$. Thus, the improvement in $\hat{L}(\mathcal{Q})$ achieved by the clustering is only about 30% while the tightness of the bound is 25%-75%. This is clearly insufficient to apply the bound as the main guideline for model order selection. However, it is possible to set the value of β in the trade-off $\mathcal{F}(\mathcal{Q}, \beta)$ via cross-validation and obtain remarkably good results. It should be pointed out that the trade-off $\mathcal{F}(\mathcal{Q}, \beta)$ was derived from the bound, thus even though the analysis is not perfectly tight it produced a useful practical result.

- Note that in the setting of this experiment the small values of m_1 and m_2 provide "natural regularization"; thus there is no significant decrease in performance when we increase β beyond 1. This will change in the following experiments.

The power of bound (18) and the trade-off $\mathcal{F}(\mathcal{Q}, \beta)$ derived from the bound is that it penalizes the effective complexity of the solution rather than the gross number of clusters used. The practical implication of this property is that we can initialize the optimization algorithm with more clusters than are actually required to solve a problem, and the algorithm will automatically adjust the extent to which it uses said available clusters. This property is verified in the following two experiments. In the first experiment we initialized Algorithm 1 with $m_1 = m_2 = 50$ clusters along each dimension. The result of optimization of $\mathcal{F}(\mathcal{Q}, \beta)$ as a function of β is presented in Figure 4. We make the following observations based on this experiment:

- The best performance (the 0.72 test MAE) achieved in the previous setting with $m_1 = 13$ and $m_2 = 6$ is achieved in the new setting with $m_1 = m_2 = 50$ as well. This supports the ability



(a) Bound (18). (b) Test Loss (zoom into subfigure a.).

Figure 5: Co-clustering of the MovieLens data set into 283x283 clusters. Figure (a) shows the value of bound (18) together with the MAE on the test set as a function of β. Figure (b) zooms into MAE on the test set. The values of β are on a log scale. See text for further details.

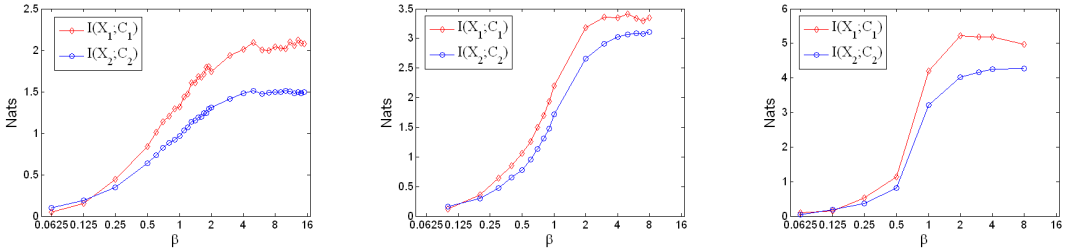
of the algorithm to operate with more clusters than are actually required by the problem and to adjust the complexity of the solution automatically.

- Note that the optimal value of β in this setting is below 1. In particular, this implies that the MDL formulation, which corresponds to $\beta = 1$ would overfit in this case. The role of the regularization parameter β is also more evidently expressed here compared to the preceding experiment.

- The values of the bound, although less tight than in the previous case, are still meaningful. The shape of the bound becomes closer to the shape of the test loss, although in light of the preceding experiment we would not attribute importance to it, and would still prefer to set the value of β via cross-validation.

In our last experiment we went to the extreme case of taking $m_1 = m_2 = 283 = \sqrt{N}$. Note that the size of the cluster space $M = m_1 m_2$ in this case is $80,089$ and is equal to the size of the training set, $N = 80,000$. The implication is that extensive use of all available clusters can result in a situation where each partition cell contains an order of a single observation, which is clearly insufficient for statistically reliable predictions. Thus, in this experiment the number of clusters provides no regularization at all and the only parameter responsible for regularization of the model is the trade-off parameter β. The result of the experiment is presented in Figure 5. We highlight the following points regarding this experiment:

- The best performance (the 0.72 test MAE) is achieved in this experiment as well. This further stresses the ability to have full control over regularization of the model via parameter β of the trade-off $\mathcal{F}(\mathcal{Q}, \beta)$.



(a) 13x6 clusters (b) 50x50 clusters (c) 283x283 clusters

Figure 6: The values of $\bar{I}(X_i;C_i)$ in the minimized $\mathcal{F}(\mathcal{Q},\beta)$ corresponding to Figures 3, 4, 5. We show the values of $\bar{I}(X_i;C_i)$ that were obtained after minimization of $\mathcal{F}(\mathcal{Q},\beta)$ by $\mathcal{Q}$ when clustering the MovieLens data set into 13x6, 50x50, and 283x283 clusters.

- The role of the regularization parameter β is further increased in this experiment compared to the previous two. The optimal value of β here is clearly below 1 (the optimal $\beta \approx 0.5$), suggesting that the MDL solution would be overfitting.

- The value of the bound still remains meaningful, although it is already quite far from the test error. The shape of the bound does not seem to provide useful information and the value of β should be set via cross-validation.

In Figure 6 we show how the mutual information $\bar{I}(X_1;C_1)$ and $\bar{I}(X_2;C_2)$ changed in the three experiments as we optimized $\mathcal{F}(\mathcal{Q},\beta)$ by $\mathcal{Q}$ for increasing values of β. An important observation to be made from these graphs (by relating them to Figures 3, 4, and 5) is that in all three experiments the best prediction performance was achieved at roughly the same values of the mutual information $\bar{I}(X_1;C_1)$ and $\bar{I}(X_2;C_2)$. For clustering into 13x6 clusters prediction performance of MAE equal to 0.72 and slightly lower was achieved at β values starting from 0.7 and larger, when $\bar{I}(X_1;C_1)$ was in the range between 1.1 and 2.1 and $\bar{I}(X_2;C_2)$ was in the range between 0.8 and 1.5; for clustering into 50x50 clusters prediction performance of 0.72 and slightly lower was achieved for β values in the range between 0.5 and 1.0, when $\bar{I}(X_1;C_1)$ was in the range between 1.1 and 2.2 and $\bar{I}(X_2;C_2)$ was in the range between 0.8 and 1.7; and for clustering into 283x283 clusters the optimal prediction performance of slightly below 0.72 was achieved for $\beta = 0.5$ and at this value of β we had $\bar{I}(X_1;C_1) = 1.1$ and $\bar{I}(X_2;C_2) = 0.8$. We see that although the three experiments were initialized with different numbers of clusters m_1 and m_2, the optimal prediction performance was achieved at roughly the same *effective complexity* of the solution (measured by $\bar{I}(X_i;C_i)$-s) and that the trade-off parameter β took care of regularization of the model.

6. Probabilistic Matrix Tri-Factorization

For $d = 2$ the discriminative prediction and density estimation models considered in this paper can be seen as two forms of matrix tri-factorization and for higher dimensions $d > 2$ as Tucker decompositions. In this section we discuss this relation in more detail, starting from the discriminative prediction problem.

6.1 Probabilistic Matrix Tri-Factorization for Discriminative Prediction

For $d = 2$ Equation (17) accepts the form:

$$q(y|x_1, x_2) = \sum_{c_1,c_2} q(c_1|x_1)q(y|c_1, c_2)q(c_2|x_2).$$

If $q(y|c_1, c_2)$ is restricted to be a delta distribution then it can be replaced by a function $f(c_1, c_2)$. This means that instead of drawing y in the cluster product space according to a distribution $q(y|c_1, c_2)$ it is predicted deterministically by $f(c_1, c_2)$. Note that the assignment of x_i-s to c_i-s remains stochastic. The restriction of deterministic $q(y|c_1, c_2)$ does not limit the model significantly, since for many loss functions, such as zero-one, absolute, or quadratic losses the optimal prediction rule is deterministic in the cluster product space in any case. At the same time the bounds derived previously are still valid, since they are valid for any distribution $q(y|c_1, c_2)$ and in particular for the delta distribution. The restricted model accepts the form:

$$f(x_1, x_2) = \sum_{c_1,c_2} q(c_1|x_1)f(c_1, c_2)q(c_2|x_2),$$

which can be written as a matrix product:

$$A = Q_1^T F Q_2, \tag{45}$$

where

$$Q_i = [q(c_i|x_i)] \qquad (i = 1, 2)$$

are $m_i \times n_i$ *left* stochastic matrices[5] mapping x_i-s to their clusters c_i-s, and

$$F = [f(c_1, c_2)]$$

is an $m_1 \times m_2$ matrix describing what happens in the cluster product space.

Given a data matrix A (probably sparse) and a trade-off parameter β Algorithm 1 provides a locally optimal approximation of A in the form of (45) regularized by the mutual information preserved in Q_1 and Q_2. Note that Algorithm 1 naturally handles the missing entries in A. The product $Q_1^T F Q_2$ can then be used to complete the missing entries.

Matrix factorization of the form (45) was already considered in Banerjee et al. (2007) without regularization. However, in Banerjee et al. (2007) the matrices Q_1 and Q_2 are restricted to deterministic assignments of x_i-s to c_i-s (the entries of Q_1 and Q_2 are in $\{0, 1\}$), whereas in the factorization proposed here Q_1 and Q_2 are stochastic matrices and F is arbitrary. Matrix tri-factorization considered in Ding et al. (2006); Yoo and Choi (2009a) is more closely related to matrix tri-factorization for density estimation discussed in the next subsection. We note that the recent Bayesian approaches to matrix factorization (Shan and Banerjee, 2008; Salakhutdinov and Mnih, 2008) are three-level stochastic models and unlike the two-level stochastic model in (45) cannot be written as a simple product of matrices. The following list of positive properties of probabilistic matrix tri-factorization suggested here further distinguishes it from other forms of matrix factorization, including singular value decomposition (SVD) (Strang, 2009; Golub and Loan, 1996), non-negative matrix factorization (Lee and Seung, 1999, 2001), low-rank matrix factorization (Srebro et al., 2005a), and maximum margin matrix factorization (Srebro et al., 2005b; Srebro, 2004), that satisfy only parts of the list:

5. A *left* stochastic matrix is a matrix of non-negative real numbers with each column summing up to 1. In a *right* stochastic matrix each row sums up to 1.

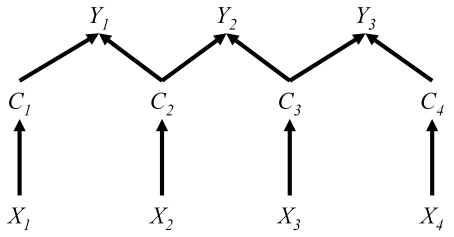


Figure 7: A graphical model for simultaneous tri-factorization of multiple matrices (equations (46)-(48)). The mapping of X_i to C_i is modeled by a corresponding matrix Q_i and labels Y_i correspond to prediction tasks in the respective matrices A_i.

- It has a clear probabilistic interpretation.
- It naturally handles missing values.
- The factorized matrix A can be arbitrary (not necessarily positive or positive definite).
- Overfitting can be controlled via the regularization parameter β.
- The generalization bound derived for co-clustering applies to this form of matrix factorization.
- It is a two-level stochastic model of the data.
- The model can be optimized by iterative projections.
- The model achieves state-of-the-art results in prediction of missing matrix entries.

We leave a wider practical comparison of the different matrix factorization methods as a subject for future work.

A promising direction for future research suggested in Seldin (2009) and independently in Yoo and Choi (2009b) is to apply matrix tri-factorization in tasks, where multiple related data sets are considered. For example, let A_1 be a matrix of viewers-by-viewers properties, A_2 be a collaborative filtering matrix, and A_3 be a matrix of movies-by-movies properties. We can look for simultaneous tri-factorizations, such that:

$$A_1 \approx Q_1^T F_1 Q_2 \tag{46}$$

$$A_2 \approx Q_2^T F_2 Q_3 \tag{47}$$

$$A_3 \approx Q_3^T F_3 Q_4. \tag{48}$$

In other words, the clustering of viewers into clusters of viewers is shared between factorizations of A_1 and A_2 and the clustering of movies into clusters of movies is shared between factorizations of A_2 and A_3. An unregularized form of simultaneous tri-factorization for collaborative filtering was already explored in Yoo and Choi (2009b). Problems of a similar form are also frequent in bioinformatics, when multiple experiments with partial relations are considered. For example, Alter et al. (2003) applied generalized SVD (GSVD) to compare yeast and human cell-cycle gene expression data sets. In their experiment it is natural to create separate systems of clusters for yeast and human

genes, but a common system of clusters for the cell-cycle time points. As already pointed out, probabilistic matrix tri-factorization suggested here has several advantages over SVD (and consequently over GSVD). Hence, it would be interesting to apply it to this type of problem.

6.2 Probabilistic Matrix Tri-Factorization for Density Estimation

The model for discrete density estimation based on co-clustering in Equation (26) can also be written as matrix tri-factorization (for $d = 2$):

$$A = R_1^T G R_2, \tag{49}$$

where

$$R_i = [q(x_i|c_i)] = \left[\frac{q(x_i)}{q(c_i)} q(c_i|x_i)\right] \qquad (i = 1, 2)$$

are $m_i \times n_i$ *right* stochastic matrices of probabilities of generating x_i-s given c_i-s and

$$G = [q(c_1, c_2)]$$

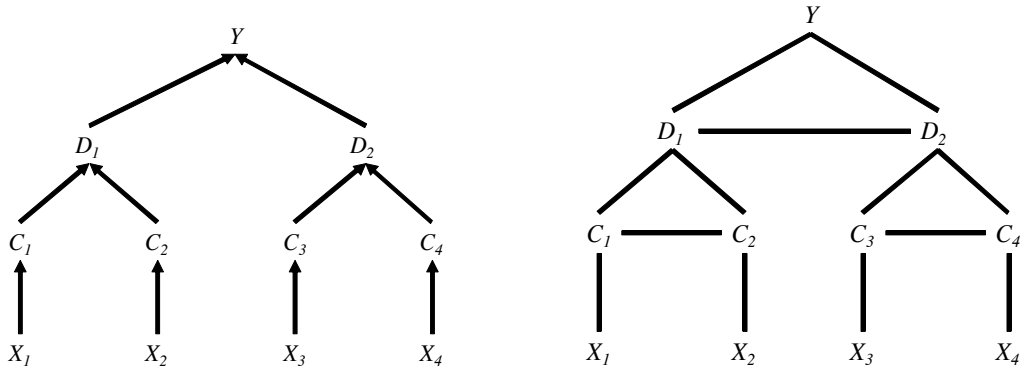
is an $m_1 \times m_2$ matrix of joint probability distribution of c_1 and c_2. As already mentioned, this model is appropriate for co-occurrence data analysis, such as word-document co-occurrence matrices. An unregularized form of such decomposition was already considered in Ding et al. (2006) and Yoo and Choi (2009a). Here, A is assumed to be a joint probability matrix (i.e., the entries of A are non-negative and sum up to 1). In practice A is an empirical joint probability distribution matrix and factorization (49) regularized by the mutual informations $\bar{I}(X_i; C_i)$ can be used to regularize the estimation of the joint probability distribution. Algorithm 2 can be used to find a local optimum of such factorization given the regularization parameter β. The generalization bound developed in Theorem 12 holds for this factorization. We remind the reader that Algorithm 2 operates with deterministic assignments of x_i-s to clusters c_i-s. Although the resulting reverse conditional distribution $q(x_i|c_i)$ is not deterministic, the algorithm does not explore all possible solutions to this problem. Developing an algorithm for finding a local optimum of the regularized decomposition with mixed memberships of x_i-s is a challenging direction for future research.

7. PAC-Bayesian Analysis of Graphical Models

The analysis of co-clustering presented in Section 3 holds for any dimension d. However, the dependence of the bounds (18), (31), and (36) on d is exponential because of the $M = \prod_i m_i$ term that they involve. This term is reasonably small when the number of dimensions is small (two or three), as in the example of co-clustering. However, as the number of dimensions grows, this term grows exponentially. Thus, high dimensional tasks require a different treatment. Some improvements are also possible if we consider discriminative prediction based on a single parameter X (i.e., in the case of $d = 1$), but the one-dimensional case is beyond the scope of this paper and we refer the interested reader to (Seldin and Tishby, 2008; Seldin, 2009) for further details. In this section we suggest a hierarchical approach to handle high-dimensional problems. We then show that this approach can also be applied to generalization analysis of graphical models.

7.1 Hierarchical Approach to High Dimensional Problems ($d > 2$)

One possible way to handle high dimensional problems is to use hierarchical partitions, as shown in Figures 8 and 9. For example, the discriminative prediction rule corresponding to the model in



(a) Discriminative Prediction Model (b) Moralization of (a)

Figure 8: Illustration of (a) a graphical model for discriminative prediction and (b) its moralized counterpart. The illustration is for $d = 4$.

Figure 8.a is:

$$q(y|x_1,..,x_4) = \sum_{d_1,d_2} q(y|d_1,d_2) \sum_{c_1,..,c_4} \prod_{i=1}^{2} q(d_i|c_{2i-1},c_{2i}) \prod_{j=1}^{4} q(c_j|x_j). \tag{50}$$

And the corresponding randomized prediction strategy is $\mathcal{Q} = \{\{q(c_i|x_i)\}_{i=1}^4, \{q(d_i|c_{2i-1},c_{2i})\}_{i=1}^2, q(y|d_1,d_2)\}$. In this case the hypothesis space is the space of all hard partitions of x_i-s to c_i-s and of the pairs $\langle c_{2i-1},c_{2i}\rangle$ to d_i-s. By repeating the analysis in Theorem 7 we obtain that with probability greater than $1-\delta$:

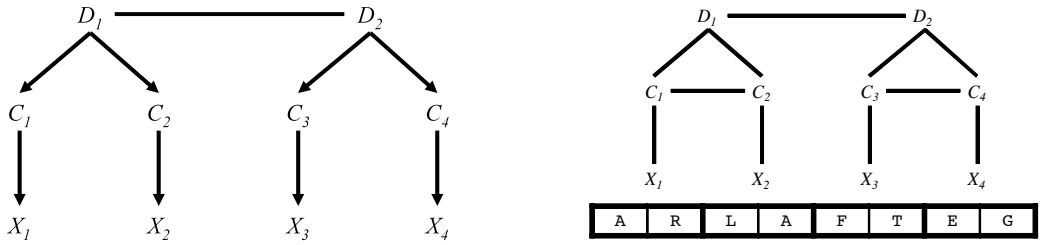
$$kl(\hat{L}(\mathcal{Q})\|L(\mathcal{Q})) \leq \frac{B_1 + B_2 + |D_1||D_2|\ln|\mathcal{Y}| + \frac{1}{2}\ln(4N) - \ln\delta}{N}, \tag{51}$$

where

$$B_1 = \sum_{i=1}^{4} \left(n_i \bar{I}(X_i;C_i) + m_i \ln n_i \right),$$
$$B_2 = \sum_{i=1}^{2} \left((m_{2i-1}m_{2i}) \bar{I}(D_i;C_{2i-1},C_{2i}) + |D_i| \ln(m_{2i-1}m_{2i}) \right).$$

Observe that the $M\ln|\mathcal{Y}|$ term in (18), which corresponds to the clique $\langle C_1,C_2,C_3,C_4,Y\rangle$, is replaced in (51) with terms which correspond to much smaller cliques $\langle C_1,C_2,D_1\rangle$, $\langle C_3,C_4,D_2\rangle$, and $\langle D_1,D_2,Y\rangle$. This factorization makes it possible to control the complexity of the partition and the tightness of the bound. In a similar way it is possible to derive factorized analogs to bounds (31) and (36) that apply to density estimation hierarchies as in Figure 9.

We provide an illustration of a possible application of the models in Figure 9. Imagine that we intend to analyze protein sequences. Protein sequences are sequences over the alphabet of 20 amino acids. Subsequences of length 8 can reach $20^8 = 256 \cdot 10^8$ instances. Instead of studying this space directly, which would require an order of $256 \cdot 10^8$ samples, we can associate each X_i



(a) Density Estimation Model (b) Moralization and Application of (a)

Figure 9: Illustration of (a) a directed model for density estimation and (b) its moralization and application to sequence modeling. The sequence in subfigure (b) is an imaginary subsequence of length 8 of a protein sequence. Each X_i corresponds to a pair of amino acids in the subsequence.

with a pair of amino acids - see Figure 9. The subspace of pairs of amino acids is only $20^2 = 400$ instances and local interactions between adjacent pairs of amino acids can easily be studied. We can cluster the pairs of amino acids into, say, 20 clusters C. Interactions between adjacent pairs of C-s in such a construction correspond to interactions between quadruples of amino acids. The subspace of quadruples is $20^4 = 16 \cdot 10^4$ instances. However, the reduced subspace of pairs of C_i-s is only $20^2 = 400$ instances. Thus, we have doubled the range of interactions, but remained at the same level of complexity. We can further cluster pairs of C_i-s (which correspond to quadruples of amino acids) into D_i-s and study the space of 8-tuples of amino acids while remaining at the same level of complexity.

The above approach shares the same basic principle already discussed in the collaborative filtering task: by clustering together similar pairs (and then quadruples) of amino acids we increase the statistical reliability of the observations, but reduce the resolution at which we process the data. Bound (51) suggests how the trade-off between model resolution and statistical reliability can be optimized.

7.2 PAC-Bayesian Analysis of Graphical Models

The result in the previous subsection suggests a new approach to learning graphical models by providing a way to evaluate the expected performance of a graphical model on new data. Thus, instead of constructing a graphical model that fits the observed data it serves to construct a model with good generalization properties. The analysis used to derive bound (51) can be applied to any directed graphical model in the form of a tree (as in Figures 8.a, 9.a) or its moralized counterpart (as in Figures 8.b, 9.b). The analysis shows that the generalization power of these graphical models is determined by a trade-off between empirical performance and the amount of mutual information that is propagated up the tree. It is important to note that the PAC-Bayesian bound is able to take advantage of the factor form of distribution (50) and that bound (51) depends on the sizes of the tree cliques, but not on the size of the parameter space $\mathcal{X}_1 \times .. \times \mathcal{X}_d$. Further, a prior can be added over all possible directed graphs under consideration to obtain a PAC-Bayesian bound that will hold for all of them simultaneously. Development of efficient algorithms for optimization of the tree structure and extension of the results to more general graphical models are key directions for future research.

8. PAC-Bayesian Analysis of Graph Clustering and Pairwise Clustering

In this section we show that our approach to predictive formulation of unsupervised learning problems and their subsequent PAC-Bayesian analysis can also be applied to weighted graph clustering (and, consequently, to pairwise clustering, which can be regarded as clustering of a graph with edge weights corresponding to pairwise distances). Graph clustering is an important tool in data analysis with a wide variety of applications including social networks analysis, bioinformatics, image processing, and many more. As a result, a multitude of different approaches to graph clustering were developed. Examples include graph cut methods (Shi and Malik, 2000), spectral clustering (Ng et al., 2001), information-theoretic approaches (Slonim et al., 2005), to name just a few. Comparing the different approaches is usually a painful task, mainly because the goal of each of these clustering methods is formulated in terms of the solution: most clustering methods start by defining some objective functional and then minimize it. But, for a given problem, how can we choose whether to apply a graph cut method, spectral clustering, or an information-theoretic approach?

Here we formulate weighted graph clustering as a prediction problem.[6] Given a subset of edge weights we analyze the ability of graph clustering to predict the remaining edge weights. The rationale behind this formulation is that if a model (not necessarily cluster-based) is able to predict with high precision all edge weights of a graph given a small subset of edge weights then it is a good model of the graph. The advantage of this formulation of graph modeling is that it is independent of the specific way chosen to model the graph and can be used to compare any two solutions, either by comparison of generalization bounds or by cross-validation. The generalization bounds or cross-validation also address the finite-sample nature of the graph clustering problem and provide a clear criterion for model order selection. For very large data sets, where computational constraints can prevent considering all edges of a graph, as for example in Yom-Tov and Slonim (2009), the generalization bounds can be used to resolve the trade-off between computational workload and precision of graph modeling.

Below we provide a PAC-Bayesian analysis of graph clustering in the case, where independent sampling of edge weights is possible. For example, in the analysis of flow graphs, such as loads on links in traffic or communication networks, we can repeatedly sample one or more non-adjacent edge weights at a time (non-adjacent edges have no common vertices). In the analysis of snapshot graphs, for example image segmentation, the dependencies between adjacent edge weight samples should be taken into account, but this is again beyond the scope of this paper.

8.1 PAC-Bayesian Analysis of Graph Clustering

Assume that $\mathcal{X}$ is a space of $|\mathcal{X}|$ nodes and denote by $w_{x_1x_2}$ the weight of an edge connecting nodes x_1 and x_2.[7] We assume that the weights $w_{x_1x_2}$ are generated according to an unknown probability distribution $p(w|x_1,x_2)$. We further assume that the space of nodes $\mathcal{X}$ is known and we are given a sample of size N of edge weights, generated according to $p(x_1,x_2,w)$. The goal is to build a regression function $q(w|x_1,x_2)$ that will minimize the expected prediction error of the edge weights $\mathbb{E}_{p(x_1,x_2,w)}\mathbb{E}_{q(w'|x_1,x_2)}l(W,W')$ for some externally given loss function $l(w,w')$. Note that this formulation does not assume any specific form of $q(w|x_1,x_2)$ and enables comparison of all possible approaches to this problem.

6. Unweighted graphs can be modeled by setting the weight of present edges as 1 and absent edges as 0.

7. All the results can be straightforwardly extended to hyper-graphs.

Here we analyze generalization abilities of $q(w|x_1,x_2)$ based on clustering:

$$q(w|x_1,x_2)=\sum_{c_1,c_2} q(w|c_1,c_2)q(c_1|x_1)q(c_2|x_2). \tag{52}$$

One can immediately see the relation between (52) and the co-clustering model for discriminative prediction (17). The only difference is that in (52) the nodes x_1,x_2 belong to the same space of nodes $\mathcal{X}$ and the conditional distribution $q(c|x)$ is shared for the mapping of endpoints of an edge. Let $\hat{p}(x_1,x_2,w)$ be the empirical distribution over edge weights. The empirical loss of prediction strategy $\mathcal{Q}=\{q(c|x),q(w|c_1,c_2)\}$ corresponding to (52) can then be written as:

$$\hat{L}(\mathcal{Q})=\mathbb{E}_{\hat{p}(x_1,x_2,w)}\mathbb{E}_{q(w'|x_1,x_2)}l(W,W').$$

The following generalization bound for graph clustering can be proved by a minor adaptation of the proof of Theorem 7.

Theorem 13 *For any probability measure $p(x_1,x_2,w)$ over the space of nodes and edge weights $\mathcal{X}\times\mathcal{X}\times\mathcal{W}$ and for any loss function l bounded by 1, with probability of at least $1-\delta$ over a selection of an i.i.d. sample S of size N according to p, for all graph clustering models defined by $\mathcal{Q}=\{q(c|x),q(w|c_1,c_2)\}$:*

$$kl(\hat{L}(\mathcal{Q})\|L(\mathcal{Q}))\leq\frac{n\bar{I}(X;C)+m\ln n+m^2\ln|\mathcal{W}|+\frac{1}{2}\ln(4N)-\ln\delta}{N}, \tag{53}$$

where $m=|C|$, $n=|\mathcal{X}|$, and $|\mathcal{W}|$ is the number of distinct edge weights.

Continuous edge weights can be handled by post-process quantization, as shown in appendix B.

As in the case of co-clustering, in practice we can replace (53) with a trade-off:

$$\mathcal{J}(\mathcal{Q},\beta)=\beta N\hat{L}(\mathcal{Q})+n\bar{I}(X;C) \tag{54}$$

and tune β either by substituting $\hat{L}(\mathcal{Q})$ and $\bar{I}(X;C)$ resulting from the solution of (54) back into the bound or via cross-validation.

If the distribution $q(w|c_1,c_2)$ is restricted to be a delta distribution Equation (52) can be rewritten as:

$$f(x_1,x_2)=\sum_{c_1,c_2} q(c_1|x_1)f(c_1,c_2)q(c_2|x_2)$$

and the corresponding matrix form is:

$$A\approx Q^TFQ,$$

where $Q=[q(c|x)]$, $F=[f(c_1,c_2)]$, and A is an input matrix (probably sparse) providing a sample of graph edge weights. This form of symmetric matrix tri-factorization is briefly mentioned in Ding et al. (2006). Let $\mathbb{I}$ be an indicator matrix for the entries present in A. For quadratic loss the empirical approximation error $\hat{L}(\mathcal{Q})$ can be written as:

$$\hat{L}(\mathcal{Q})=\frac{1}{N}\|\mathbb{I}\circ(A-Q^TFQ)\|_2^2, \tag{55}$$

where $\circ$ denotes entrywise product of two matrices. From (55) it is easy to see that $\hat{L}(\mathcal{Q})$ is not convex in Q (unlike in the case of co-clustering, where the dependence on Q_i-s was quadratic,

here the power of Q is 4). Hence, although alternating projections similar to those in Algorithm 1 can be applied to $\mathcal{J}(\mathcal{Q},\beta)$ they are not guaranteed to converge to a local minimum. Nevertheless, according to some preliminary experiments they can achieve reasonably good solutions and bound (53) provides reasonably tight guarantees on the expected approximation quality (Seldin, 2010). Alternatively, trade-off $\mathcal{J}(\mathcal{Q},\beta)$ can be optimized by sequential minimization. Unlike alternating projections, sequential minimization (similar to the one in Algorithm 2) is guaranteed to converge to a local minimum, but can operate with deterministic assignments of graph nodes x_i-s to the clusters c_i-s only. A convergent algorithm that will be able to explore stochastic assignments still awaits to be developed.

8.2 Related Work on Pairwise Clustering

The regularization of pairwise clustering by mutual information $\bar{I}(X;C)$ was already applied in practice by Slonim et al. (2005). They maximized a parameterized trade-off $\beta\langle s\rangle - \bar{I}(X;C)$, where $\langle s\rangle = \sum_c \bar{q}(c) \sum_{x_1,x_2} q(x_1|c)q(x_2|c)w_{x_1x_2}$ measured average pairwise similarities within a cluster.[8] Their algorithm demonstrated superior results in cluster coherence compared to 18 other clustering methods. The regularization by mutual information was motivated by information-theoretic considerations inspired by the rate distortion theory (Cover and Thomas, 1991). Namely, the authors drew a parallel between $\langle s\rangle$ and distortion and $\bar{I}(X;C)$ and compression rate of a clustering algorithm. Further, Yom-Tov and Slonim (2009) showed that the algorithm can be run in parallel mode, where each parallel worker operates with a subset of pairwise relations at each iteration rather than all of them. Such a mode of operation was motivated by inability to consider all pairwise relations in very large data sets due to computational constraints. Yom-Tov and Slonim (2009) reported only minor empirical degradation in clustering quality, but no formal analysis or guarantees were suggested. The results presented here can help to analyze such problems and help to address the trade-off between computational workload and approximation quality in analysis of very large graphs.

9. Related Work on Clustering

The idea of considering clustering in the context of a higher level task was inspired by the Information Bottleneck (IB) principle (Tishby et al., 1999; Slonim, 2002; Slonim et al., 2006). The IB principle considers the problem of extracting information from a random variable X that is relevant for prediction of a random variable Y. The relevance variable Y defines the high-level task. For example, X might be a speech signal and the task might be identification of the speaker or transcription of the signal. The extraction of relevant information from X is done by means of clustering of $\mathcal{X}$ into clusters $\tilde{\mathcal{X}}$ that preserve the information on $\mathcal{Y}$ (Tishby et al., 1999). Clearly, each relevance variable $\mathcal{Y}$ corresponds to a different partition (clustering) of $\mathcal{X}$. The IB principle was further extended to graphical models in Slonim et al. (2006).

The idea to consider clustering as a proxy to solution of a prediction task was further developed in Krupka and Tishby (2005, 2008) and Krupka (2008). Krupka and Tishby analyze a scenario wherein each object has multiple properties, but only a fraction of the properties is observed. Consider the following illustration: assume we are presented with multiple fruits and we observe their parameters, such as size, color, and weight. We can cluster the fruits by their observed parameters

8. The loss $L(\mathcal{Q})$ is slightly more general than $\langle s\rangle$ since it also considers edges between the clusters. Although, this generality breaks the convexity of $\hat{L}(\mathcal{Q})$ in (55).

in order to facilitate prediction of unobserved parameters, such as taste and toxicity. This approach enables one to conduct a formal analysis and derive generalization bounds for prediction rules based on clustering.

In recent years extensive attempts have been made to address the question of model order selection in clustering through evaluation of its stability (Lange et al., 2004; von Luxburg and Ben-David, 2005; Ben-David et al., 2006; Shamir and Tishby, 2009; Ben-David and von Luxburg, 2008). This perspective suggests that for two random samples generated by the same source, clustering of the samples should be similar (and hence stable). Otherwise the obtained clustering is unreliable. Although it has been proven that in a large sample regime stability can be used for model order selection (Shamir and Tishby, 2009), no upper bounds on the minimal sample size required for stability estimates to hold can be proved. Moreover, in certain cases stability indices based on arbitrarily large samples can be misleading (Ben-David and von Luxburg, 2008). Since in any practical application the amount of data available is limited, currently existing stability indices cannot be used for reliable model order selection and it is not clear whether the stability indices can be used to compare solutions based on different optimization objectives.

Gaussian ring example. We use the following example from Seldin (2009) to illustrate that generalization and stability criteria for evaluation of clustering are not equivalent. Assume points in $\mathbb{R}^2$ are generated according to the following process. First, we select a center μ of a Gaussian according to a uniform distribution on a unit circle in $\mathbb{R}^2$. Then we generate a point $x \sim \mathcal{N}(\mu, \sigma^2 I)$ according to a Gaussian distribution centered at μ with a covariance matrix $\sigma^2 I$ for a fixed σ (I is a 2 by 2 identity matrix). Given a sample generated according to the above process we can apply a mixture of Gaussians clustering in order to learn the generating distribution. Note that:

1. Due to the circular symmetry in the generating process and model redundancy, the solution will always be unstable (the centers of Gaussians in the mixture of Gaussians model can move arbitrarily along the unit circle and their variance in the direction tangential to the circle is loosely constrained).

2. By increasing the sample size and the number of Gaussians in the mixture of Gaussians model the true density of the points can be approximated arbitrarily well.

Hence, models with good generalization properties are not necessarily stable. This point should be kept in mind when using generalization as an evaluation criterion in clustering.

10. Discussion and Future Work

This paper underlines the importance of external evaluation of unsupervised learning, such as clustering or more general structure learning, based on the context of its potential application. Such a form of evaluation is important for delivery of better structure learning algorithms as well as for better understanding of their outcome. We argue that structure learning does not occur for its own sake, but rather in order to facilitate a solution of some higher level goal. In any non-trivial data many structures co-exist simultaneously and it is a matter of the subsequent usage of the outcome of the learning algorithm to determine which of the structure elements are valuable and which are not. Therefore, unsupervised learning cannot be analyzed in isolation from its potential application. In our opinion, one of the main obstacles in theoretical advancement of unsupervised learning is an absence of a good mathematical definition of the context of its application. The analysis of co-clustering presented here is a first step toward context-based analysis of more complex models.

The work presented here started with an attempt to improve our understanding of clustering. We note that clustering is tightly related to the object naming process in human language. In a sense, a cluster is an entity that can be assigned a name. By clustering objects we ignore their irrelevant properties and concentrate on the relevant ones. And of course, this division can change according to our needs. For example, we can divide animals into birds and mammals or into flying and notatorial or into domestic and wild. Whereas the classification into birds and mammals or flying and notatorial may be considered intrinsic, the classification into domestic and wild is definitely application-oriented. In order to design successful clustering and categorization algorithms it is important to understand the basic principles behind this process. It is not a-priori clear that, if we restrict ourselves to pure prediction tasks, clustering the underlying sample space helps. As shown in Seldin and Tishby (2008); Seldin (2009), in classification by a single parameter there is no need to cluster the parameter space, but rather simple smoothing performs better. In classification in higher dimensional spaces, kernel-based methods can be superior to clustering-based approaches. However, we know that as humans we communicate by using a clustered representation of the world rather than by kernel matrices. Thus, there should be advantages for such form of communication. Identification, understanding, and analysis of these advantages is an important future direction for the design of better clustering and higher structure learning algorithms.

Acknowledgments

We are very grateful to the anonymous reviewers for their valuable comments that helped us to improve this manuscript significantly. We would also like to thank Ohad Shamir, Joachim M. Buhmann, and Suvrit Sra for helpful discussions. This work was supported in part by the Israel Science Foundation Bikura grant 942/09 and the NATO SfP BESAFE CBM.MD.SFPP-982480 grant.

Appendix A. Proof of Theorem 6

Proof of Theorem 6 First we prove inequality (14):

$$\begin{aligned}
-\mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{p_h(z)}\ln\tilde{p}_h(Z) &= \mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{[\hat{p}_h(z)-p_h(z)]}\ln\tilde{p}_h(Z) - \mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{\hat{p}_h(z)}\ln\tilde{p}_h(Z) \\
&= \mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{[\hat{p}_h(z)-p_h(z)]}\ln\frac{\hat{p}_h(Z)+\gamma}{1+\gamma|\mathcal{Z}|} - \mathbb{E}_{\mathcal{Q}(h)}\mathbb{E}_{\hat{p}_h(z)}\ln\frac{\hat{p}_h(Z)+\gamma}{1+\gamma|\mathcal{Z}|} \\
&\leq -\frac{1}{2}\|\hat{p}_{\mathcal{Q}}(z)-p_{\mathcal{Q}}(z)\|_1\ln\frac{\gamma}{1+\gamma|\mathcal{Z}|} + \mathbb{E}_{\mathcal{Q}(h)}H(\hat{p}_h(z)) + \ln(1+\gamma|\mathcal{Z}|) \\
&\leq H(\hat{p}_{\mathcal{Q}}(z)) - \sqrt{\varepsilon/2}\ln\frac{\gamma}{1+\gamma|\mathcal{Z}|} + \ln(1+\gamma|\mathcal{Z}|). \qquad (56)
\end{aligned}$$

The last inequality is justified by the concavity of the entropy function H and the KL-divergence bound on the L_1 norm (Cover and Thomas, 1991):

$$\|\hat{p}_{\mathcal{Q}}(z)-p_{\mathcal{Q}}(z)\|_1 \leq \sqrt{2KL(\hat{p}_{\mathcal{Q}}(z)\|p_{\mathcal{Q}}(z))} \leq \sqrt{2\varepsilon}.$$

By differentiation (56) is minimized by $\gamma = \frac{\sqrt{\varepsilon/2}}{|\mathcal{Z}|}$. By substitution of this value of γ into (56) we obtain (14). Inequality (15) is justified by (14) and the concavity of the $\ln$ function. Finally, we

prove the lower bound (16):

$$\begin{aligned}-\mathbb{E}_{p_{\mathcal{Q}}(z)} \ln \tilde{p}_{\mathcal{Q}}(Z) &= \mathbb{E}_{[\hat{p}_{\mathcal{Q}}(z)-p_{\mathcal{Q}}(z)]} \ln \tilde{p}_{\mathcal{Q}}(Z) - \mathbb{E}_{\hat{p}_{\mathcal{Q}}(z)} \ln \tilde{p}_{\mathcal{Q}}(Z) \\ &\geq -\frac{1}{2}\|\hat{p}_{\mathcal{Q}}(z) - p_{\mathcal{Q}}(z)\|_1 \ln \frac{1+\gamma|\mathcal{Z}|}{\gamma} + H(\hat{p}_{\mathcal{Q}}(z)) \\ &\geq H(\hat{p}_{\mathcal{Q}}(z)) - \sqrt{\varepsilon/2} \ln \frac{|\mathcal{Z}|(1+\sqrt{\varepsilon/2})}{\sqrt{\varepsilon/2}}.\end{aligned}$$

■

Appendix B. Treatment of Continuous Label Spaces $\mathcal{Y}$ via Quantization

In Theorems 7 and 13 it was assumed that the label space $\mathcal{Y}$ (or the edge weight space $\mathcal{W}$) is finite. However, for quadratic loss the minimization of $\mathcal{F}(\mathcal{Q}, \beta)$ by Algorithm 1 can return a solution that falls out of this finite space. Furthermore, the input space $\mathcal{Y}$ itself does not have to be finite (e.g., gene expression levels in bioinformatics can be given on a continuous scale). Here we show that the bound can be easily generalized to handle this case via quantization of $\mathcal{Y}$. The analysis can be seen as post-processing and does not require modifications of the trade-off $\mathcal{F}(\mathcal{Q}, \beta)$ and of Algorithm 1, since the algorithm does not assume finiteness of $\mathcal{Y}$.

Assume that $\mathcal{Y}$ is limited in [0,1] interval and apply uniform quantization of $\mathcal{Y}$ at intervals Δ, then $|\mathcal{Y}_\Delta| = \frac{1}{\Delta}$ ($\mathcal{Y}_\Delta$ is the quantized copy of $\mathcal{Y}$ and we assume that the quantization starts at $\frac{1}{2}\Delta$ and ends at $1 - \frac{1}{2}\Delta$). By rounding the continuous values of y obtained by Algorithm 1 toward the closest quantization both the empirical and the expected loss are increased by at most $2\Delta + \Delta^2$ (in the case of quadratic loss). This is because quantization can shift the true y and the prediction y' by at most $\frac{1}{2}\Delta$ and then $l(y - \frac{1}{2}\Delta, y' + \frac{1}{2}\Delta) = (y - y' - \Delta)^2 = (y - y')^2 - 2(y - y')\Delta + \Delta^2 \leq l(y, y') + 2\Delta + \Delta^2$, where the last inequality follows from the assumption that $\mathcal{Y}$ is limited in [0,1]. Hence, we have

$$L(\mathcal{Q}) \leq kl^{-1}\left(\hat{L}(\mathcal{Q}) + 2\Delta + \Delta^2, \frac{\sum_{i=1}^{d} n_i \bar{I}(X_i; C_i) + K}{N}\right) + 2\Delta + \Delta^2,$$

where K, defined previously in (38), becomes:

$$K = \sum_{i=1}^{d} m_i \ln n_i - M \ln \Delta + \frac{1}{2}\ln(4N) - \ln \delta.$$

As a rule of thumb one can choose $\Delta = kM/N$ for $k \approx 5$, so that the contribution of Δ to the two operands of the inverse KL-divergence is approximately equivalent. In general this correction for quantization has no significant influence on the bound (Seldin, 2010).

References

Orly Alter, Patrick O. Brown, and David Botstein. Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms. In *Proceedings of the National Academy of Science*, 2003.

Amiran Ambroladze, Emilio Parrado-Hernández, and John Shawe-Taylor. Tighter PAC-Bayes bounds. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

Surugu Arimoto. An algorithm for computing the capacity of discrete memoryless channel. *IEEE Transactions on Information Theory*, 18, 1972.

Jean-Yves Audibert and Olivier Bousquet. Combining PAC-Bayesian and generic chaining bounds. *Journal of Machine Learning Research*, 2007.

Arindam Banerjee. On Bayesian bounds. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2006.

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, Srujana Merugu, and Dhamendra S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8, 2007.

Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, 2001.

Peter L. Bartlett, Stéphane Boucheron, and Gábor Lugosi. Model selection and error estimation. *Machine Learning*, 2001.

Peter L. Bartlett, Michael Collins, Ben Taskar, and David McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

Shai Ben-David and Ulrike von Luxburg. Relating clustering stability to properties of cluster boundaries. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, 2008.

Shai Ben-David, Ulrike von Luxburg, and David Pál. A sober look on clustering stability. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

Richard E. Blahut. Computation of channel capacity and rate distortion functions. *IEEE Transactions on Information Theory*, 18, 1972.

Gilles Blanchard and François Fleuret. Occam's hammer. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, 2007.

David Blei and John Lafferty. Topic models. In A. Srivastava and M. Sahami, editors, *Text Mining: Theory and Applications*. Taylor and Francis, 2009.

Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Theory of classification: a survey of recent advances. *ESAIM: Probability and Statistics*, 2005.

Olivier Catoni. PAC-Bayesian supervised classification: The thermodynamics of statistical learning. *IMS Lecture Notes Monograph Series*, 56, 2007.

Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the 8^{th} International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 2000.

Hyuk Cho and Inderjit S. Dhillon. Co-clustering of human cancer microarrays using minimum sum-squared residue co-clustering. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 5(3), 2008.

Hyuk Cho, Inderjit S. Dhillon, Yuqiang Guan, and Suvrit Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, 2004.

Thomas M. Cover. Admissibility properties of Gilberts encoding for unknown source probabilities. *IEEE Transactions on Information Theory*, 18, 1972.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

Koby Crammer, Mehryar Mohri, and Fernando Pereira. Gaussian margin machines. In *Proceedings on the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Philip Derbeko, Ran El-Yaniv, and Ron Meir. Explicit learning curves for transduction and application to clustering and compression algorithms. *Journal of Artificial Intelligence Research*, 22, 2004.

Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, 2003.

Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, 2006.

Miroslav Dudík, Steven J. Phillips, and Robert E. Schapire. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, 8, 2007.

Ran El-Yaniv and Oren Souroujon. Iterative double clustering for unsupervised and semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

Dayane Freitag. Trained named entity recognition using distributional clusters. In *Proceedings of EMNLP*, 2004.

Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM05)*, 2005.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009.

Edgar N. Gilbert. Codes based on inaccurate source probabilities. *IEEE Transactions on Information Theory*, 17(3), 1971.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3^{rd} edition, 1996.

Peter Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

Isabelle Guyon, Ulrike von Luxburg, and Robert C. Williamson. Clustering: Science or art? towards principled approaches. nips workshop, 2009.

John A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337), 1972.

Jonathan Herlocker, Joseph Konstan, Loren Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1), 2004.

Matthew Higgs and John Shawe-Taylor. A PAC-Bayes bound for tailored density estimation. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT)*, 2010.

Edwin Thompson Jaynes. Information theory and statistical mechanics. *Physical Review*, 106, 1957.

Michael Kearns, Yishay Mansour, Andrew Ng, and Dana Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning*, 1997.

Yong-Deok Kim and Seungjin Choi. Nonnegative Tucker decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

Yuval Kluger, Ronen Basri, Joseph T. Chang, and Mark Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Research*, 2003.

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 2001.

Rafail E. Krichevskiy. Laplaces law of succession and universal encoding. *IEEE Transactions on Information Theory*, 44(1), 1998.

Eyal Krupka. *Generalization from Observed to Unobserved Features*. PhD thesis, The Hebrew University of Jerusalem, 2008.

Eyal Krupka and Naftali Tishby. Generalization in clustering with unobserved features. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

Eyal Krupka and Naftali Tishby. Generalization from observed to unobserved features by clustering. *Journal of Machine Learning Research*, 9, 2008.

Tilman Lange, Volker Roth, Mikio L. Braun, and Joachim M. Buhmann. Stability based validation of clustering solutions. *Neural Computation*, 2004.

John Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 2005.

John Langford and John Shawe-Taylor. PAC-Bayes & margins. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

Danial Lashkari and Polina Golland. Co-clustering with generative models. Technical report, MIT-CSAIL-TR-2009-054, 2009.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401, 1999.

Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

Guy Lever, François Laviolette, and John Shawe-Taylor. Distribution-dependent PAC-Bayes priors. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT)*, 2010.

Hang Li and Naoki Abe. Word clustering and disambiguation based on co-occurrence data. In *Proceedings of the 17th International Conference on Computational Linguistics*, 1998.

Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1), 2004.

Yishay Mansour and David McAllester. Generalization bounds for decision trees. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, 2000.

Andreas Maurer. A note on the PAC-Bayesian theorem. www.arxiv.org, 2004.

David McAllester. Some PAC-Bayesian theorems. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, 1998.

David McAllester. PAC-Bayesian model averaging. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, 1999.

David McAllester. Simplified PAC-Bayesian margin bounds. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, 2003.

David McAllester. Generalization bounds and consistency for structured labeling. In Gökhan Bakir, Thomas Hofmann, Bernhard Schölkopf, Alexander Smola, Ben Taskar, and S.V.N. Vishwanathan, editors, *Predicting Structured Data*. The MIT Press, 2007.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

Liam Paninski. Estimation of entropy and mutual information. *Neural Computation*, 2003.

Liam Paninski. Variational minimax estimation of discrete distributions under kl loss. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

Richard Rohwer and Dayne Freitag. Towards full automation of lexicon construction. In Dan Moldovan and Roxana Girju, editors, *HLT-NAACL 2004: Workshop on Computational Lexical Semantics*, 2004.

Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain monte carlo. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

Matthias Seeger. PAC-Bayesian generalization error bounds for Gaussian process classification. *Journal of Machine Learning Research*, 2002.

Matthias Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalization Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, 2003.

Yevgeny Seldin. *A PAC-Bayesian Approach to Structure Learning*. PhD thesis, The Hebrew University of Jerusalem, 2009.

Yevgeny Seldin. A PAC-Bayesian analysis of graph clustering and pairwise clustering. Technical report, http://arxiv.org/abs/1009.0499, 2010.

Yevgeny Seldin and Naftali Tishby. Multi-classification by categorical features via clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

Yevgeny Seldin and Naftali Tishby. PAC-Bayesian generalization bound for density estimation with application to co-clustering. In *Proceedings on the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Yevgeny Seldin, Noam Slonim, and Naftali Tishby. Information bottleneck for non co-occurrence data. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

M. Mahdi Shafiei and Evangelos E. Milios. Model-based overlapping co-clustering. In *Proceeding of SIAM Conference on Data Mining*, 2006.

Ohad Shamir and Naftali Tishby. On the reliability of clustering stability in the large sample regime. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

Hanhuai Shan and Arindam Banerjee. Bayesian co-clustering. In *IEEE International Conference on Data Mining (ICDM)*, 2008.

John Shawe-Taylor and David Hardoon. Pac-bayes analysis of maximum entropy classification. In *Proceedings on the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

John Shawe-Taylor and Robert C. Williamson. A PAC analysis of a Bayesian estimator. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, 1997.

John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5), 1998.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

Noam Slonim. *The Information Bottleneck: Theory and Applications*. PhD thesis, The Hebrew University of Jerusalem, 2002.

Noam Slonim and Naftali Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

Noam Slonim, Nir Friedman, and Naftali Tishby. Unsupervised document classification using sequential information maximization. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

Noam Slonim, Gurinder Singh Atwal, Gasper Tracik, and William Bialek. Information-based clustering. *Proceedings of the National Academy of Science*, 102(51), 2005.

Noam Slonim, Nir Friedman, and Naftali Tishby. Multivariate information bottleneck. *Neural Computation*, 18, 2006.

Nathan Srebro. *Learning with Matrix Factorizations*. PhD thesis, MIT, 2004.

Nathan Srebro, Noga Alon, and Tommi S. Jaakkola. Generalization error bounds for collaborative prediction with low-rank matrices. In *Advances in Neural Information Processing Systems (NIPS)*, 2005a.

Nathan Srebro, Jason Rennie, and Tommi Jaakkola. Maximum margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2005b.

Mark Steyvers and Tom Griffiths. Probabilistic topic models. In T. Landauer, D. McNamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum, 2006.

Gilbert Strang. *Introduction to linear algebra*. Wellesley-Cambridge Press, 4^{th} edition, 2009.

Ilya Sutskever, Ruslan Salakhutdinov, and Josh B. Tenenbaum. Modelling relational data using Bayesian clustered tensor factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

Hiroya Takamura and Yuji Matsumoto. Co-clustering for text categorization. *Information Processing Society of Japan Journal*, 2003.

Naftali Tishby, Fernando Pereira, and William Bialek. The information bottleneck method. In *Allerton Conference on Communication, Control and Computation*, 1999.

Leslie G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11), 1984.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Soviet Math. Dokl.*, 9, 1968.

Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2), 1971.

Ulrike von Luxburg and Shai Ben-David. Towards a statistical theory of clustering. In *PASCAL Workshop on Statistics and Optimization of Clustering*, 2005.

Pu Wang, Carlotta Domeniconi, and Kathryn Blackmond Laskey. Latent Dirichlet Bayesian co-clustering. In *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2009.

Elad Yom-Tov and Noam Slonim. Parallel pairwise clustering. In *SIAM International Conference on Data Mining (SDM)*, 2009.

Jiho Yoo and Seungjin Choi. Probabilistic matrix tri-factorization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2009a.

Jiho Yoo and Seungjin Choi. Weighted nonnegative matrix co-tri-factorization for collaborative prediction. In *Proceedings of the Asian Conference on Machine Learning (ACML)*, 2009b.

Journal of Machine Learning Research 11 (2010) 3647-3680 Submitted 6/08; Revised 1/10; Published 12/10

Learning Non-Stationary Dynamic Bayesian Networks

Joshua W. Robinson JOSH@CS.DUKE.EDU
Alexander J. Hartemink AMINK@CS.DUKE.EDU
Department of Computer Science
Duke University
Durham, NC 27708-0129, USA

Editor: Zoubin Ghahramani

Abstract

Learning dynamic Bayesian network structures provides a principled mechanism for identifying conditional dependencies in time-series data. An important assumption of traditional DBN structure learning is that the data are generated by a stationary process, an assumption that is not true in many important settings. In this paper, we introduce a new class of graphical model called a non-stationary dynamic Bayesian network, in which the conditional dependence structure of the underlying data-generation process is permitted to change over time. Non-stationary dynamic Bayesian networks represent a new framework for studying problems in which the structure of a network is evolving over time. Some examples of evolving networks are transcriptional regulatory networks during an organism's development, neural pathways during learning, and traffic patterns during the day. We define the non-stationary DBN model, present an MCMC sampling algorithm for learning the structure of the model from time-series data under different assumptions, and demonstrate the effectiveness of the algorithm on both simulated and biological data.

Keywords: Bayesian networks, graphical models, model selection, structure learning, Monte Carlo methods

1. Introduction

A principled mechanism for identifying conditional dependencies in time-series data is provided through structure learning of dynamic Bayesian networks. An important requirement of this learning is that the time-series data is generated by a distribution that does not change with time—it is stationary. The assumption of stationarity is adequate in many situations since certain aspects of data acquisition or generation can be easily controlled and repeated. However, other interesting and important circumstances exist where that assumption does not hold and potential non-stationarity cannot be ignored.

The inspiration for this model of "networks evolving over time" comes primarily from neurobiology. Dynamic Bayesian networks have been used to identify networks of neural information flow that operate in the brains of songbirds (Smith et al., 2006). When a juvenile male songbird is born, he cannot sing any songs; however, as he ages, he learns songs from other birds singing around him. As the songbird learns from its environment, the networks of neural information flow slowly adapt to make the processing of sensory information more efficient. The analysis carried out by Smith et al. (2006) was limited by the fact that the researchers were forced to learn networks on subsets of the data since DBN structure learning algorithms assume stationarity of the data. Therefore, a learn-

ing algorithm based upon dynamic Bayesian networks that relaxes the data stationarity assumption would be ideally suited to this problem.

As another example, structure learning of DBNs has been used widely in reconstructing transcriptional regulatory networks from gene expression data (Friedman et al., 2000; Hartemink et al., 2001). But during development, these regulatory networks are evolving over time, with certain conditional dependencies between gene products being created as the organism develops, and others being destroyed. As yet another example, one can use a DBN to model traffic flow patterns. The roads upon which traffic passes do not change on a daily basis, but the dynamic utilization of those roads changes daily during morning rush, lunch, evening rush, and weekends.

If one collects time-series data describing the levels of gene products in the case of transcriptional regulation, traffic density in the case of traffic flow, or neural activity in the case of neural information flow, and attempts to learn a DBN describing the conditional dependencies in the time-series, one could be seriously misled if the data-generation process is non-stationary.

Here, we introduce a new class of graphical model called a non-stationary dynamic Bayesian network (nsDBN), in which the conditional dependence structure of the underlying data-generation process is permitted to change over time. In the remainder of the paper, we introduce and define the nsDBN framework, present a simple but elegant algorithm for efficiently learning the structure of an nsDBN from time-series data under a variety of different assumptions, and demonstrate the effectiveness of these algorithms on both simulated and experimental data.

1.1 Related Work

Representing relationships or statistical dependencies between variables in the form of a network is a popular technique in many communities, from economics to computational biology to sociology. Recently, researchers have been interested in elucidating the temporal evolution of genetic regulatory networks (Arbeitman et al., 2002; Luscombe et al., 2004) and neural information flow networks (Smith et al., 2006), but were forced to perform their analysis on subsets of the data since their structure learning algorithms assumed stationarity of the data. Additionally, some economics techniques require prior specification of a graphical model (for a particular data set) and assume it is stationary (Carvalho and West, 2007) when the data set may actually be highly non-stationary (Xuan and Murphy, 2007). Therefore, the identification of non-stationary behavior in graphical models is of significant interest and importance to many communities.

We divide models of evolving statistical dependencies into those that model changing structures and those that model changing parameters, and describe examples of both in this section. Additionally, we briefly explain how our approach compares and some of the advantages it provides over other methods.

1.1.1 STRUCTURAL NON-STATIONARITY

Approaches that learn structural non-stationarities are those that explicitly model the presence of statistical dependencies between variables and allow them to appear and disappear over time (e.g., they define directed or undirected networks whose edges change over time).

In recent work modeling the temporal progression of networks from the social networks community (Hanneke and Xing, 2006), the $p*$ or exponential random graph model (ERGM) (Wasserman and Pattison, 1996) was generalized to the temporal ERGM (tERGM) model, where a structural evolutionary process is modeled with a set of features between adjacent network structures (Han-

neke and Xing, 2006). The tERGM model was further extended to the hidden tERGM (htERGM) to handle situations where the networks are latent (unobserved) variables that generate observed time-series data (Guo et al., 2006, 2007).

While the transition model of an htERGM allows for nearly unlimited generality in characterizing how the network structure changes with time, it is restricted to functions of temporally adjacent network structures. Therefore, an evolutionary process that differs between early observations and later observations may not be effectively captured by a single transition model. Also, the emission model defined in Guo et al. (2007) must be estimated *a priori* and only captures pairwise correlations between variables; more complicated relationships between multiple variables that change over time may be missed altogether. Finally, Guo and colleagues focus on identifying undirected edges. Although it is possible to adapt the ERGM model for directed graphs, it becomes more difficult to define the parameters of the tERGM and the emission model assuming directed edges.

In the continuous domain, some research has focused on learning the structure of a time-varying undirected Gaussian graphical model (Talih and Hengartner, 2005). These authors use a reversible-jump MCMC approach to estimate the time-varying variance structure of the data. They explicitly model the network's edges as non-zeroes in the precision matrix. While this model allows for fast, efficient sampling, it only does so by defining several restrictions to the model space. First, the structural evolutionary process is piecewise-stationary and restricted to single edge changes. Rapid and significant structural changes would be approximated by a slowly changing network structure, resulting in an inaccurate portrayal of the true evolutionary behavior of the data-generating process. Additionally, the total number of segments or epochs in the piecewise stationary process is assumed known *a priori*, thereby limiting application of the method in situations where the number of epochs is not known. Finally, this approach only identifies undirected edges (correlations), while time-series data should allow one to identify directed edges (conditional dependencies).

A similar algorithm—also based on undirected Gaussian graphical models—has been developed to segment multivariate time-series data (Xuan and Murphy, 2007). This approach iterates between a convex optimization for determining the graph structure and a dynamic programming algorithm for calculating the segmentation. This results in some notable advantages over Talih and Hengartner (2005): it has no single edge change restriction and the number of segments is calculated *a posteriori*. The main restriction, however, is that the graph structure must be decomposable. Additionally, because this method models structure as non-zeroes in the precision matrix, it only identifies undirected edges. Finally, the networks (precision matrices) in each segment are assumed independent, preventing the sharing of parameters and data between segments.

Another recently proposed model for identifying evolving networks is the temporally smoothed l_1-regularized logistic regression (TESLA) method, described by Kolar et al. (2010) and Ahmed and Xing (2009). This approach involves learning a discrete binary Markov random field with sparse, varying parameters. A useful advantage of this model is that can be reparameterized to account for either smoothly varying or abrupt sudden changes to the network structure. Additionally, it scales well to thousands of nodes. However, it is undirected, requires binary data, and only captures pairwise relationships between variables. The binary input and pairwise relationship requirements may be relaxed, but the likelihood function would have to be significantly modified to incorporate cliques; the resulting optimization problem might become intractable.

1.1.2 PARAMETER NON-STATIONARITY

Approaches that learn parameter non-stationarities are those that explicitly model the evolution of parameters over time. Here we only focus on a few that can be represented as networks with temporally changing parameters.

Switching state-space models (SSMs) represent a piecewise-stationary extension of linear dynamic systems (Ghahramani and Hinton, 2000). In an SSM, a sequence of observations is modeled as a function of several independent hidden variables which is itself controlled by a switch variable. The hidden variables as well as the switch variable all have Markovian dynamics. While this approach is similar to ours in that it describes the evolution of a piecewise-stationary process, it does have some notable differences. Our model has no hidden variables, only observed variables. Critically, our variables are not assumed to be independent; rather, their dependencies are unknown and must be estimated *a posteriori*. Additionally, the piecewise-stationary process in our approach does not follow Markovian dynamics, like the switch variable in an SSM.

A more closely related model is the recently published non-homogeneous Bayesian network (Grzegorczyk et al., 2008). In this model, the conditional distributions of the variables are assumed to follow a mixture of Gaussians. Each observation is allocated to a single mixture component where the parameters and number of the mixture components are determined *a posteriori* using an allocation sampler. Unfortunately, an allocation sampler does not allow data to be shared across different mixture components since they are assumed independent. However, this model seems to do a good job of capturing the non-stationary behavior of parameters in a Gaussian Bayesian network, assuming that the underlying structure is inferred correctly.

1.1.3 OUR APPROACH

Although we provide more detail in Section 3, for the purpose of comparison, we define our approach as the identification of a discrete Bayesian network that evolves according to a piecewise-stationary process where edges are gained and lost over time. Building on the Bayesian network model allows us to identify directed networks and results in efficient learning (under certain assumptions). Additionally, when the conditional probability distributions of the Bayesian network are multinomial, we can identify linear, non-linear, and combinatorial interactions between variables. Finally, the piecewise-stationary assumption (and additional constraints on how and when edge changes occur) allows our method to scale to large data sets with many variables and provides a natural parameterization for placing priors on the structural evolution process.

Our method falls into the category of models that identify non-stationarities in structure, not parameters. In the rest of this paper, we define non-stationarities as times at which conditional dependencies between variables are gained or lost (i.e., edges are gained or lost). We have chosen to focus on structural non-stationarity for several reasons. First, we are not as interested in making predictions about future data (such as might be the case with spam prediction) as we are in the analysis of collected data to identify non-stationary statistical relationships between variables in multivariate time-series. Additionally, the problems we analyze in this paper are highly multimodal in the posterior over structures and likely to be even more varied in the posterior over parameters. By assuming conjugate parameter priors, we address both problems by averaging out all possible parameters and only examining the posterior over structures.

However, we recognize that the parameters of the conditional distributions may also change with time. Some changes to the parameters of the conditional distributions may effectively result in

a structural change, while other changes may be dramatic, yet not alter the structure. Ultimately, a trade-off must be made between simplifying model assumptions resulting in greater statistical power versus a completely general framework requiring approximation schemes. Under our modeling assumptions, we can identify non-stationarities in the parameters of the conditional distributions that are significant enough to result in structural changes; we assume other changes are small enough to not alter edges in the predicted structure.

2. Brief Review of Structure Learning of Bayesian Networks

Bayesian networks are directed acyclic graphical models that represent conditional dependencies between variables as edges. They define a simple decomposition of the complete joint distribution—a variable x_i is conditionally independent of its non-descendants given its parents. Therefore, the joint distribution of every variable x_i can be rewritten as $P(x_1,\ldots,x_n) = \prod_i P(x_i|\pi_i)$, where π_i are the parents of x_i. Bayesian networks may be learned on time-series data, but the semantics are slightly different, leading to the dynamic Bayesian network (DBN) model. DBNs enable cyclic dependencies between variables to be modeled across time. DBNs are a special case of Bayesian networks, with modeling assumptions regarding how far back in time one variable can depend on another (minimum and maximum lag), and constraints placed on edges so that they do not go backwards in time. For notational simplicity, we assume hereafter that the minimum and maximum lag are both 1.

The task of inferring the structure (i.e., the set of conditional dependencies) of a Bayesian network is typically expressed using Bayes' rule, where the posterior probability of a given network structure G (i.e., the set of conditional dependencies) after having observed data D is given by

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)}.$$

Since $P(D)$ is the same for all structures, we see that $P(G|D) \propto P(D|G)P(G)$. The prior over networks $P(G)$ can be used to incorporate prior knowledge about the existence of specific edges (Bernard and Hartemink, 2005) or the overall topology of the network (e.g., sparse); often, prior knowledge is not available and $P(\Theta)$ is assumed uniform. The marginal likelihood $P(D|G)$ is further defined in terms of the parameters Θ_i of the conditional probability distributions (CPDs) between a variable and its parents $P(x_i|\pi_i,\Theta_i)$. The entire set of parameters Θ_i for all variables is simply Θ.

$$P(D|G) = \int P(D|\Theta,G)P(\Theta|G)\mathrm{d}\Theta. \tag{1}$$

One might be interested in inferring the values of Θ given a particular network, but we will be focusing on learning the network itself, or the set of conditional dependencies. In computer science, this task is often referred to as structure learning; in statistics, it is often called model selection. More detailed reviews of structure learning of Bayesian networks can be found in Buntine (1996), Chrisman (1998), Krause (1998), and Murphy (2001).

For the remainder of this section, we review the most common approaches to inferring dynamic Bayesian networks, thereby setting the stage for what changes we will need to make in the following section to handle inference in the non-stationary setting. For brevity, we will focus on dynamic Bayesian networks with discrete variables, though the ideas could also be applied to networks of continuous variables. Details specific to structure learning of Bayesian networks with continuous variables can be found in Hofmann and Tresp (1995) and Margaritis (2005).

2.1 Evaluating the Marginal Likelihood $P(D|G)$

Evaluation of the marginal likelihood in Equation (1) can be performed either approximately or exactly. The marginal likelihood can be approximated with the Akaike information criterion (AIC), the Bayesian information criterion (BIC) (Friedman et al., 1998), or the minimum description length (MDL) (Lam and Bacchus, 1994; Suzuki, 1996) metric. Each of these metrics suggests how model complexity should be penalized. For example, the AIC metric penalizes free parameters less strongly than the BIC metric; therefore, a Bayesian network learned using the AIC metric would be likely be more dense than a Bayesian network learned using the BIC metric.

Alternatively, if a conjugate prior is placed on Θ which is then marginalized out, the value of $P(D|G)$ can be computed exactly. Assuming that Θ parameterizes multinomially distributed conditional probability distributions, one can place a conjugate Dirichlet prior on the parameters Θ and then marginalize them out to obtain the Bayesian-Dirichlet (BD) metric. The BD metric can be further modified so that all of the networks that represent the same set of conditional independence relationships have the same probability; this is called the Bayesian-Dirichlet equivalent (BDe) metric (Heckerman et al., 1995). By using a Dirichlet prior and marginalizing over all values of Θ, the BD and BDe metrics inherently penalize more complex models, so a prior on the network $P(G)$ may not always be necessary. The primary advantage of the BD family of metrics is that the evaluation of $P(D|G)$ is exact and can be computed in closed form. However, if the assumption that the parameters of CPDs are multinomially distributed is incorrect, these metrics might not find the true network of conditional dependencies.

Since we will be modifying it later in this paper, we show the closed-form expression for the BDe metric below:

$$P(D|G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})} \tag{2}$$

where q_i is the number of configurations of the parent set π_i, r_i is the number of discrete states of variable x_i, N_{ijk} is the number of times x_i took on the value k given the parent configuration j, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, and α_{ij} and α_{ijk} are Dirichlet hyper-parameters on various entries in Θ. If α_{ijk} is set to $\alpha/(q_i r_i)$ (essentially a uniform Dirichlet prior), we get a special case of the BDe metric: the uniform BDe metric (BDeu), whose parameter priors are all controlled by a single hyperparameter α.

2.2 Deciding Between Search or Sampling Strategies

Once a form of the marginal likelihood $P(D|G)$ is defined and a method for evaluating it is chosen, one must decide whether the objective is to identify the best network or to capture the uncertainty over the space of all posterior networks. Search methods may be used to find the best network or set of networks (i.e., a mode in the space of all networks), while sampling methods may be used to estimate posterior quantities from the space of all networks.

Search methods may be exact or heuristic, but exact search for Bayesian networks is computationally infeasible for more than about 30 variables because the number of possible networks is super-exponential in the number of variables. In fact, identifying the highest scoring Bayesian network has been shown to be NP-Hard (Chickering et al., 1994). If the maximum number of parents for any variable is limited to some constant p_{max}, the total number of valid networks becomes exponential in the number of variables; however, finding the best network is still NP-Hard (for $p_{max} \geq 2$). Therefore, heuristic searches are often employed, including greedy hill-climbing,

simulated annealing (Heckerman et al., 1995), the K2 algorithm (Chickering et al., 1995), genetic algorithms (Larrañaga et al., 1996), and even ant colony optimization (de Campos et al., 2002). Most heuristic searches perform well in a variety of settings, with greedy hill-climbing and simulated annealing being the most commonly used techniques.

A different approach from search is the use of sampling techniques (Madigan et al., 1995; Giudici et al., 1999; Tarantola, 2004). If the best network is all that is desired, heuristic searches will typically find it more quickly than sampling techniques. However, sampling methods allow the probability or importance of individual edges to be evaluated over all possible networks. In settings where many modes are expected, sampling techniques will more accurately capture posterior probabilities regarding various properties of the network. The primary disadvantage of sampling methods in comparison to search methods is that they often take longer before accurate results become available.

A common sampling technique often used in this setting is the Metropolis-Hastings algorithm, which is a Markov chain Monte Carlo (MCMC) method. The M-H acceptance probability for moving from state x to state x' is shown below, where each *state* is a DBN.

$$\alpha(x,x') = \min\left\{1, \frac{p(D|x')}{p(D|x)} \times \frac{p(x' \to x)}{p(x \to x')}\right\} = \min\left\{1, \underbrace{\frac{p(D|x')}{p(D|x)}}_{\textit{likelihood ratio}} \times \underbrace{\frac{\sum_{M'} p(M')p(x|x',M')}{\sum_{M} p(M)p(x'|x,M)}}_{\textit{proposal ratio}}\right\} \quad (3)$$

where M is the move type that allows for a transition from state x to x' and M' is the reverse move type for a transition from state x' back to state x. While multiple moves may result in a transition from state x to state x' (and vice versa), typically there is only a single move for each transition. In such a case, the sums over M and M' each only include one term, and the proposal ratio can be split into two terms: one is the ratio of the proposal probabilities for move types and the other is the ratio of selecting a particular state given the current state and the move type. The choice of scoring metric determines the likelihoods, and $p(M')$ and $p(M)$ are often chosen *a priori* to be equivalent or simple to calculate.

2.3 Determining the Move Set

Once a search or sampling strategy has been selected, one must determine how to move through the space of all networks. A *move set* defines a set of local traversal operators for moving from a particular state (i.e., a network) to nearby states. The set of states that can be reached from the current one is often called the *local neighborhood* of that state. The values of $p(x'|x,M)$ and $p(x|x',M)$ in the proposal ratio are defined by the move set.

Ideally, the move set includes changes that allow posterior modes to be frequently visited. For example, it is reasonable to assume that networks that differ by a single edge will have similar likelihoods. A well-designed move set results in fast convergence since less time is spent in the low probability regions of the state space. For example, with traditional Bayesian networks, Madigan et al. (1995) proposed that the move set be: *add an edge* and *delete an edge*. However, it was later shown by Giudici and Castelo (2003) that the convergence rate could be increased with the addition of another move to the move set: *reverse an edge*.

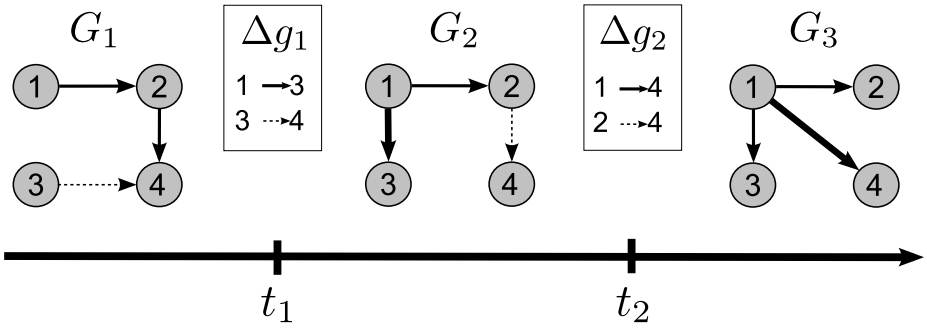


Figure 1: An example nsDBN with labeled components (namely, transition times t_1 and t_2 and edge change sets Δg_1 and Δg_2). Edges that are gained from the previous epoch are shown as thicker lines and edges that will be lost in the next epoch are shown as dashed lines. Note that the network at each epoch is actually a DBN drawn in compact form, where each edge represents a statistical dependence between a node at time t and its parent at the previous time $t-1$.

3. Learning Non-Stationary Dynamic Bayesian Networks

While DBNs are excellent models for describing dependencies between time-series random variables, they cannot represent or reason about how these dependencies might change over time. In contrast, our nsDBN model is capable of characterizing dependencies between temporally observed variables, as well as reasoning about whether and how those dependencies change. Because DBNs are well studied and well understood, we have chosen to introduce the details of our nsDBN model by building upon the existing DBN model. Therefore, we use this section to detail how the structure learning procedure for DBNs needs to be modified and extended to account for non-stationarity when learning a non-stationary DBN.

Assume that we observe the state of n random variables at N discrete times. Call this multivariate time-series data D, and further assume that it is generated according to a non-stationary process, which is unknown. The process is non-stationary in the sense that the network of conditional dependencies prevailing at any given time is itself changing over time. We call the initial (dynamic) network of conditional dependencies G_1 and subsequent networks are called $G_2, G_3, \ldots, G_m$. We define Δg_i to be the set of edges that change (either added or deleted) between G_i and G_{i+1}. The number of edge changes specified in Δg_i is s_i. We define the *transition time* t_i to be the time at which G_i is replaced by G_{i+1} in the data-generation process. We call the period of time between consecutive transition times—during which a single network of conditional dependencies is operative—an *epoch*. So we say that G_1 prevails during the first epoch, G_2 prevails during the second epoch, and so forth. We will refer to the entire series of prevailing networks as the *structure* of the nsDBN. Figure 1 shows an example nsDBN with the components labeled.

3.1 Updating the Marginal Likelihood and Incorporating Priors

Since we wish to estimate a set of networks instead of one network we must derive a new expression for the marginal likelihood. Consider the simplest case where $m = 2$ and the transition time t_1 at which the structure of the nsDBN evolves from network G_1 to network G_2 is known. We would like

to find the networks G_1 and G_2 that have the highest probability given the observed time-series data D and prior. Thus, we want to find the networks that maximize the expression below:

$$P(G_1, G_2|D, t_1) = \frac{P(D|G_1, G_2, t_1)P(G_1, G_2|t_1)}{P(D|t_1)} \propto P(D|G_1, G_2, t_1)P(G_1, G_2|t_1).$$

To maximize the marginal likelihood $P(D|G_1, G_2, t_1)$ in the above expression, one approach might be to estimate a different network for each epoch. Unfortunately, if the number of observations in each epoch is small, accurate reconstruction of the correct structure may be difficult or impossible (Friedman and Yakhini, 1996; Smith et al., 2003). Additionally, learning each network separately might lead to predictions that are vastly different during each epoch. If prior knowledge about the problem dictates that the networks will not vary dramatically across adjacent epochs, information about the networks learned in adjacent epochs can be leveraged to increase the accuracy of the network learned in the current epoch.

Expanding the simple formulation to multiple epochs, assume there exist m different epochs with transition times $T = \{t_1, \ldots, t_{m-1}\}$. The network G_{i+1} prevailing in epoch $i+1$ differs from network G_i prevailing in epoch i by a set of edge changes we call Δg_i. We would like to determine the sequence of networks $G_1, \ldots, G_m$ that maximize the posterior given below:

$$P(G_1, \ldots, G_m|D, T) \propto P(D|G_1, \ldots, G_m, T)P(G_1, \ldots, G_m|T). \tag{4}$$

Since each network differs from the previous one by a set of edge changes, we can rewrite the prior and obtain the expression below:

$$P(G_1, \ldots, G_m|T) = P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}|T).$$

By writing the objective function this way, we rephrase the problem as finding the initial network and $m-1$ sets of edge changes that best describe the data. Unfortunately, the search space for finding the best structure is still super-exponential (in n) for the initial network and the task of identifying the other networks is exponential (in both m and n). However, using this formulation, it is easy to place some constraints on the search space to reduce the number of valid structures. If we specify that the maximum number of parents for any variable is p_{max} and, through priors, require that the maximum number of edge changes for any Δg_i is s_{max}, the search space for finding the best structure becomes exponential (in n) for the initial network and exponential (in m and s_{max}) for the other networks.

We assume the prior over networks can be further split into components describing the initial network and subsequent edge changes, shown below:

$$P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}|T) = P(G_1)P(\Delta g_1, \ldots, \Delta g_{m-1}|T).$$

leading to the final form of the posterior

$$P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}|D, T) \propto P(D|G_1, \Delta g_1, \ldots, \Delta g_{m-1}, T)P(G_1)P(\Delta g_1, \ldots, \Delta g_{m-1}|T). \tag{5}$$

As in the stationary setting, if prior knowledge about particular edges or overall topology is available, an informative prior can be placed on $P(G_1)$. In the context of the simulations and experiments in this paper, we had no prior knowledge about the network; thus, we assumed a uniform prior on $P(G_1)$. We do, however, place some prior assumptions on the ways in which edges change in the

structure. First, we assume that the networks evolve smoothly over time. To encode this prior knowledge, we place a truncated geometric prior with parameter $p = 1 - e^{-\lambda_s}$ on the number of changes (s_i) in each edge change set (Δg_i), with the distribution truncated for $s_i > s_{max}$. For the problems we explore, the truncation has no noticeable effect since s_{max} is chosen to be large. The updated posterior for the structure is given below:

$$\begin{aligned} P(G_1, \Delta g_1, \ldots, \Delta g_{m-1} | D, T) &\propto P(D|G_1, \ldots, G_m, T) \prod_i \frac{\left(1 - e^{-\lambda_s}\right)\left(e^{-\lambda_s}\right)^{s_i}}{1 - \left(e^{-\lambda_s}\right)^{s_{max}+1}} \\ &\propto P(D|G_1, \ldots, G_m, T) \prod_i \left(e^{-\lambda_s}\right)^{s_i} \\ &\propto P(D|G_1, \ldots, G_m, T) e^{-\lambda_s s}, \end{aligned}$$

where $s = \sum_i s_i$. Therefore, a (truncated) geometric prior on each s_i is essentially equivalent to a (truncated) geometric prior on the sufficient statistic s, the total number of edge changes.

When the transition times are *a priori* unknown, we would like to estimate them. The posterior in this setting becomes

$$\begin{aligned} P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}, T | D) &\propto P(D|G_1, \ldots, G_m, T) P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}, T) \\ &= P(D|G_1, \ldots, G_m, T) P(G_1) P(\Delta g_1, \ldots, \Delta g_{m-1}, T) \\ &= P(D|G_1, \ldots, G_m, T) P(G_1) P(\Delta g_1, \ldots, \Delta g_{m-1}) P(T). \end{aligned}$$

We assume that the joint prior over the evolutionary behavior of the network and the locations of transition times can be decomposed into two independent components. We continue to use the previous geometric prior with parameter $p = 1 - e^{-\lambda_s}$ on the total number of edge changes. Any choice for $P(T)$ can be made, but for the purposes of this paper, we have no prior knowledge about the transition times; therefore, we assume a uniform prior on T so that all settings of transition times are equally likely for a given value of m.

Finally, when neither the transition times nor the number of epochs are known *a priori*, both the number and times of transitions must be estimated *a posteriori*. If prior knowledge dictates that the networks evolve slowly over time (i.e., a transition does not occur at every observation), we can include this knowledge by placing a non-uniform prior on m, for example a (truncated) geometric prior with parameter $p = 1 - e^{\lambda_m}$ on the number of epochs m, with the distribution truncated for $m > N$. A geometric prior on the number of epochs is equivalent to a geometric prior on epoch lengths (see Appendix A).

The form of the geometric prior was chosen for convenience since under a log transformation, the likelihood (and prior) calculations reduce to: $\log(\text{likelihood}) - \lambda_s s - \lambda_m m$. In general, large values of λ_m encode the strong prior belief that the structure changes slowly (i.e., few epochs exist) and large values of λ_s encode the strong prior belief that the structure changes smoothly (i.e., few edge changes exist).

Fortunately, the likelihood component of the posterior does not change whether we know the transition times *a priori* or not. Therefore, any uncertainty about the transition times can be incorporated into the evaluation of the prior, leaving the evaluation of the likelihood unchanged.

3.2 Evaluating the New Marginal Likelihood

Now that a new likelihood has been defined, we must decide on which method to use for evaluation and how to modify it to account for multiple structures. Any of the previously mentioned metrics

can be modified to account for non-stationarity, but we chose to extend the BDe metric because it is exact and because edges are the only representation of conditional dependencies that are left after the parameters have been marginalized away. This provides a useful paradigm for non-stationarity that is both simple to define and easy to analyze.

The BDe metric needs to be modified when learning an nsDBN because in Equation (2), N_{ij} and N_{ijk} are calculated for a particular parent set over the entire data set D. However, in an nsDBN, a node may have multiple parent sets operative at different times. The calculation for N_{ij} and N_{ijk} must therefore be modified to specify the *intervals* during which each parent set is operative. Note that an interval may include several epochs. An epoch is defined between adjacent transition times while an interval is defined as the union of consecutive epochs during which a particular parent set is operative (which may include all epochs).

For each node i, the parent set π_i in the BDe metric is replaced by a set of parent sets π_{ih}, where h indexes the interval I_h during which parent set π_{ih} is operative for node i. Letting p_i be the number of such intervals and q_{ih} be the number of configurations of π_{ih} results in the expression below:

$$P(D|G_1,\ldots,G_m,T) \propto \prod_{i=1}^{n}\prod_{h=1}^{p_i}\prod_{j=1}^{q_{ih}} \frac{\Gamma(\alpha_{ij}(I_h))}{\Gamma(\alpha_{ij}(I_h)+N_{ij}(I_h))}\prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}(I_h)+N_{ijk}(I_h))}{\Gamma(\alpha_{ijk}(I_h))}$$

where the counts N_{ijk} and pseudocounts α_{ijk} have been modified to apply only to the data in each interval I_h. The modified BDe metric will be referred to as nsBDe. Given that $|I_h| = t_{h+1} - t_h$, we set $\alpha_{ijk}(I_h) = (\alpha_{ijk}|I_h|)/N$ (e.g., proportional to the length of the interval during which that particular parent set is operative). If α_{ijk} is set everywhere to $\alpha/(q_i r_i)$ as in the BDeu metric, then we generalize the BDeu metric to nsBDeu, and the parameter priors are again all controlled by a single hyperparameter α.

The derivation of the BDe metric requires seven assumptions: multinomial sample, parameter independence, likelihood modularity, parameter modularity, Dirichlet distribution for parameters, complete data, likelihood equivalence, and structure possibility (Heckerman et al., 1995). To extend the BDe metric to account for non-stationary behavior, we must extend only the parameter independence assumption.

Parameter independence is split into local parameter independence and global parameter independence. Letting G represent an individual network, global parameter independence is represented as $p(\Theta_G|G) = \prod_{i=1}^{n} p(\Theta_i|G)$. In other words, the conditional probabilities can be decomposed by variable. Local parameter independence is represented as $p(\Theta_i|G) = \prod_{j=1}^{q_i} p(\Theta_{ij}|G)$; the conditional probabilities for each variable are decomposable by parent configuration.

For nsDBNs, we also need to assume parameter independence across intervals. Parameter independence is thus split into three assumptions. The updated parameter independence assumptions are defined below, where $\mathbb{G}$ represents an nsDBN structure, or set of networks.

Global parameter independence: The conditional probabilities are decomposable by variable:

$$p(\Theta_{\mathbb{G}}|\mathbb{G}) = \prod_{i=1}^{n} p(\Theta_i|\mathbb{G}).$$

Interval parameter independence: The conditional probabilities for each variable are decomposable by interval:

$$p(\Theta_i|\mathbb{G}) = \prod_{h=1}^{p_i} p(\Theta_{ih}|\mathbb{G}).$$

Local parameter independence: The conditional probabilities for each variable for each interval are decomposable by parent configuration:

$$p(\Theta_{ih}|\mathbb{G}) = \prod_{j=1}^{q_{ih}} p(\Theta_{ihj}|\mathbb{G}).$$

3.3 Using a Sampling Strategy and Choosing a New Move Set

We decide to take a sampling approach rather than using heuristic search because the posterior over structures includes many modes. In particular, when the transition times are not known *a priori*, the posterior is highly multimodal because structures with slightly different transition times likely have similar posterior probabilities. Additionally, sampling offers the further benefit of allowing us to evaluate interesting posterior quantities, such as when are the most likely times at which transitions occur—a question that would be difficult to answer in the context of heuristic search.

Because the number of possible nsDBN structures is so large (significantly greater than the number of possible DBNs), we must be careful about what is included in the move set. To provide quick convergence, we want to ensure that every move in the move set efficiently jumps between posterior modes. Therefore, the majority of the next section is devoted to describing effective move sets under different levels of uncertainty.

4. Settings

Each of the following subsections demonstrates a method for calculating an nsDBN under a variety of *settings* that differ in terms of the level of uncertainty about the number and times of transitions. The different settings will be abbreviated according to the type of uncertainty: whether the *number* of transitions is *known* (KN) or *unknown* (UN) and whether the transition *times* themselves are *known* (KT) or *unknown* (UT). Figure 2 shows plate diagrams relating the DBN model to the three different settings of the nsDBN model, described in the following subsections.

4.1 Known Number and Known Times of Transitions (KNKT)

In the KNKT setting, we know all of the transition times *a priori*; therefore, we only need to identify the most likely initial network G_1 and sets of edge changes $\Delta g_1, \ldots, \Delta g_{m-1}$. Thus, we wish to maximize Equation (5).

To create a move set that results in an effectively mixing chain, we consider which types of local moves result in jumps between posterior modes. As mentioned earlier, networks that differ by a single edge will probably have similar likelihoods. Therefore, the move set includes a single edge addition or deletion to G_1. Each of these moves results in the structural difference of a single edge over all observations. One can also consider adding or deleting an edge in a particular Δg_i; this results in the structural difference of a single edge for all observations after t_i. Finally, we consider moving an edge from one Δg_i to another, which results in the structural difference of a single edge for all observations between t_i and t_j. These moves are listed as M_1–M_5 in Table 1, along with the various proposal probabilities defined in the Metropolis-Hastings acceptance ratio shown in Equation (3).

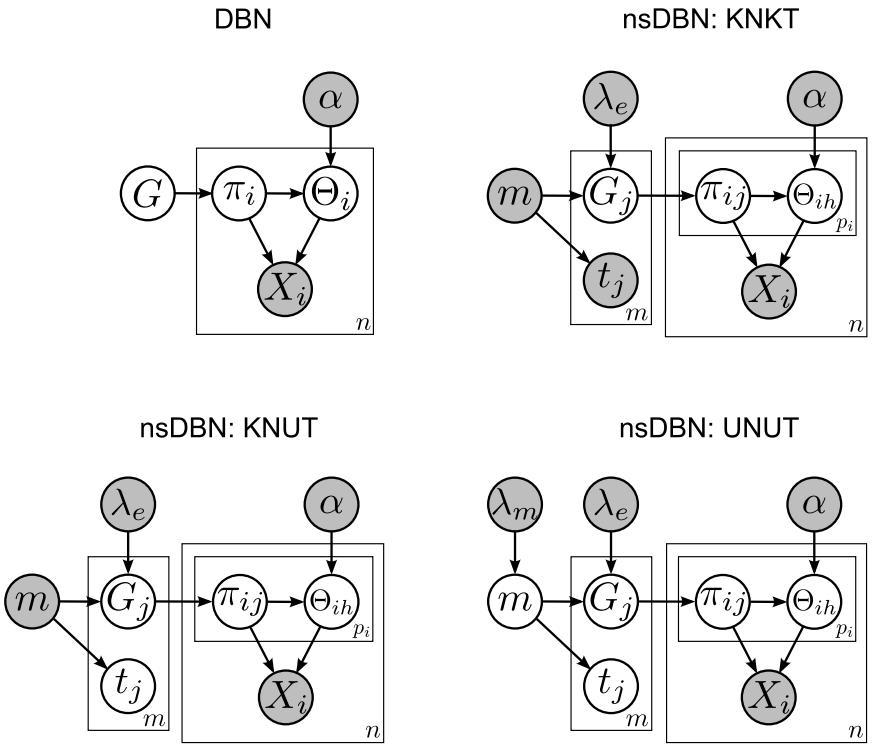


Figure 2: Plate diagrams relating DBNs to nsDBNs under each setting. Gray circles represent quantities that are known *a priori* while white circles represent those that are unknown. Non-stationary DBNs may have multiple networks (m, indexed by j) and multiple different parent sets for each variable (p_i for variable i, indexed by h). In the KNKT setting, the transition times t_j are known *a priori*, but they must be estimated in the two other settings. In the KNUT setting, the number of epochs m is still known, but the transition times themselves are not. Finally, in the UNUT setting, even the number of epochs is unknown; instead a truncated geometric prior is placed on m.

4.2 Known Number But Unknown Times of Transitions (KNUT)

Knowing in advance the times at which all the transitions occur, as was assumed in the previous subsection, is often unrealistic. To relax this assumption, we now assume that while m is known, the set T is not given *a priori* but must also be estimated. Thus, rather than maximizing Equation (5), we maximize the expression below:

$$P(G_1, \Delta g_1, \ldots, \Delta g_{m-1}, T|D).$$

Structures with the same edge sets but slightly different transition times will probably have similar likelihoods. Therefore, we can add a new move that proposes a local shift to one of the transition times: let d be some small positive integer and let the new time t_i' be drawn from a discrete uniform distribution $t_i' \sim DU(t_i - d, t_i + d)$ with the constraint that $t_{i-1} < t_i' < t_{i+1}$. Initially, we set the $m-1$ transition times so that the epochs are roughly equal in length. This placement allows the transition times ample room to locally shift without "bumping" into each other too early in the sampling

procedure. The complete move set for this setting includes all of the moves described previously as well as this new *local shift move*, listed as M_6 in Table 1.

As with the last setting, the number of epochs does not change; therefore, only the prior on the number of edge changes s is used.

4.3 Unknown Number and Unknown Times of Transitions (UNUT)

In the most general UNUT setting, both the transition times T and the number of transitions are unknown and must be estimated. While this is the most interesting setting, it is also the most difficult. Since the move set from the KNUT setting provides a solution to this problem when m is known, a simple approach would be to try various values of m and then determine which value of m seems optimal. However, this approach is theoretically unsatisfying and would be incredibly slow. Instead, we will further augment the move set to allow the number of transitions to change. Since both the number of edge changes s and the number of epochs m are allowed to vary, we need to incorporate both priors mentioned in Section 3.1 when evaluating the posterior.

To allow the number of epochs m to change during sampling, we introduce *merge* and *split* operations to the move set. For the merge operation, two adjacent edge sets (Δg_i and Δg_{i+1}) are combined to create a new edge set. The transition time of the new edge set is selected to be the mean of the previous locations weighted by the size of each edge set: $t_i' = (s_i t_i + s_{i+1} t_{i+1})/(s_i + s_{i+1})$. For the split operation, an edge set Δg_i is randomly chosen and randomly partitioned into two new edge sets $\Delta g_i'$ and $\Delta g_{i+1}'$ with all subsequent edge sets re-indexed appropriately. Each new transition time is selected as described above. The move set is completed with the inclusion of the *add transition time* and *delete transition time* operations. These moves are similar to the split and merge operations except they also increase or decrease s, the total number of edge changes in the structure. The four additional moves are listed as M_7–M_{10} in Table 1.

4.4 MCMC Sampler Implementation Details

In practice, the sampler is designed so that the proposal ratio $\frac{p(M')}{p(M)} \frac{p(x|x',M')}{p(x'|x,M)}$ is exactly 1 for most moves. For example, if either move M_1 or move M_2 is randomly selected, the sampling procedure is as follows: random variables x_i and x_j are selected, if the edge $x_i \to x_j$ exists in G_1, it is deleted, otherwise it is added (subject to the maximum of p_{max} parents constraint). We know that the maximal number of edges in G_1 is np_{max} (due to the maximum parent constraint) and we let E_1 be the current number of edges in G_1. If we are making move M_1, the probability that we select a legal edge to add is $P_a = \frac{np_{max}-E_1}{np_{max}}$. The probability of making the reverse move (from x' back to x) is $P_d = \frac{E_1+1}{np_{max}}$. The resulting proposal ratio is thus:

$$\frac{P_d}{P_a} \frac{p(x|x',M_2')}{p(x'|x,M_1)} = \frac{E_1+1}{np_{max}-E_1} \frac{np_{max}-E_1}{E_1+1} = 1.$$

A similar approach can be applied to the other moves.

This paradigm also handles boundary cases: if G_1 is complete, then $P(M_1) = 0$; if G_1 is empty, then $P(M_2) = 0$; if $s_i = s_{max} \forall i$, then $P(M_3) = 0$; if $s_i = 1 \forall i$, then $P(M_4) = 0$; etc.

The relative proposal probabilities between different moves are designed so that all pairs of complementary moves (a move and its reverse move) are equally likely. Under the KNKT setting, $P_a + P_d = P_{ae} + P_{de} = 2P_{me}$. Under the KNUT setting, $P_a + P_d = P_{ae} + P_{de} = 2P_{me} = 2P_{st}$. Finally, under the UNUT setting, $P_a + P_d = P_{ae} + P_{de} = 2P_{me} = 2P_{st} = P_m + P_s = P_{ag} + P_{dg}$.

Move type M	Proposal probability	$\frac{p(M')}{p(M)}$	$\frac{p(x\|x',M')}{p(x'\|x,M)}$	
(M_1) add edge to G_1	P_a	$\frac{P_d}{P_a}$	$\frac{(E_1+1)^{-1}}{(np_{max}-E_1)^{-1}} = \frac{np_{max}-E_1}{E_1+1}$	KNKT, KNUT, UNUT
(M_2) delete edge from G_1	P_d	$\frac{P_a}{P_d}$	$\frac{(np_{max}-E_1+1)^{-1}}{E_1^{-1}} = \frac{E_1}{np_{max}-E_1+1}$	KNKT, KNUT, UNUT
(M_3) add edge to Δg_i	P_{ae}	$\frac{P_{de}}{P_{ae}}$	$\frac{m^{-1}(S_i+1)^{-1}}{m^{-1}(S_{max}-S_i)^{-1}} = \frac{S_{max}-S_i}{S_i+1}$	KNKT, KNUT, UNUT
(M_4) delete edge from Δg_i	P_{de}	$\frac{P_{ae}}{P_{de}}$	$\frac{m^{-1}(S_{max}-S_i+1)^{-1}}{m^{-1}S_i^{-1}} = \frac{S_i}{S_{max}-S_i+1}$	KNKT, KNUT, UNUT
(M_5) move edge from Δg_i to Δg_j	P_{me}	1	$\frac{(m-1)^{-1}(\sum_i S_i)^{-1}}{(m-1)^{-1}(\sum_i S_i)^{-1}} = 1$	KNKT, KNUT, UNUT
(M_6) locally shift t_i	P_{st}	1	$\frac{(2d+1)^{-1}}{(2d+1)^{-1}} = 1$	KNUT, UNUT
(M_7) merge Δg_i and Δg_{i+1}	P_m	$\frac{P_s}{P_m}$	$\frac{(m-1)\mathbb{1}^1 2(S_i+S_{i+1})^{-1}\binom{S_i+S_{i+1}}{S_i}^{-1}}{(m-1)^{-1}} = \frac{2}{(S_i+S_{i+1})\binom{S_i+S_{i+1}}{S_i}}$	KNUT, UNUT
(M_8) split Δg_i	P_s	$\frac{P_m}{P_s}$	$\frac{(m-1)^{-1}}{(m-1)^{-1}(S_i/2)^{-1}\binom{S_i}{x}^{-1}} = (S_i/2)\binom{S_i}{x}$	KNUT, UNUT
(M_9) create new Δg_i	P_{ag}	$\frac{P_{dg}}{P_{ag}}$	$\frac{(m+1)^{-1}}{(N-m)^{-1}n^{-2}} = \frac{(N-m)n^2}{m+1}$	UNUT
(M_{10}) delete Δg_i	P_{dg}	$\frac{P_{ag}}{P_{dg}}$	$\frac{(N-m-1)^{-1}n^{-2}}{m^{-1}} = \frac{m}{(N-m-1)n^2}$	UNUT

Table 1: The move sets under different settings. E_1 is the total number of edges in G_1, p_{max} is the maximum parent set size, s_{max} is the maximum number of edge changes allowed in a single transition time, and s_i is the number of edge changes in the set Δg_i. The proposal ratio is the product of the last two columns. The KNKT setting uses moves M_1–M_5, KNUT uses moves M_1–M_6, and UNUT uses moves M_1–M_{10}, in each case with the proposal probabilities appropriately normalized to add to 1.

5. Results on Simulated and Real Data Sets

Here we examine both the speed and accuracy of our sampling algorithm under all three settings and on both simulated and real data sets. We want to solve real-world problems, but accurate ground truths are often not available to assess performance; therefore, we must rely on simulation studies to provide representative performance estimates for the real problems of interest.

We have studied the performance characteristics of our algorithm in simulation studies that vary by several orders of magnitude in number of observations, number of variables, number of epochs, and network density. In each case, we perform simulations with multiple data sets multiple times with multiple chains to help ensure our results are robust to simulation artifacts. For brevity, we only present a few simulation results here; the broader set of experiments we have conducted yield similar results.

All experiments were run on a 3.6GHz dual-core Intel Xeon machine with 4 GB of RAM.

5.1 Small Simulated Data Set

To evaluate the effectiveness of our method, we first apply it to a small, simulated data set. The first experiment is on a simulated ten node network with 1020 observations and six single-edge changes between seven epochs, where the length of each epoch varies between 20 and 400 observations. The

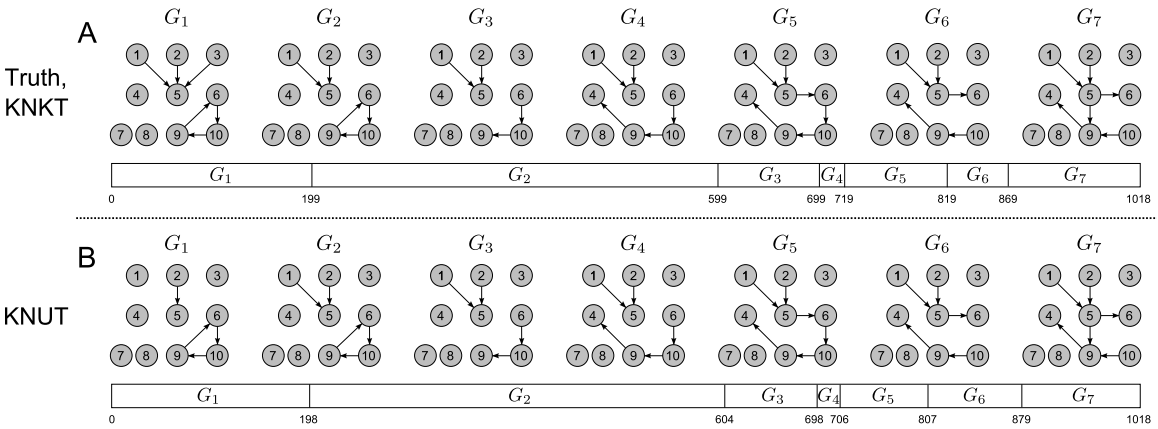


Figure 3: nsDBN structure learning with known numbers of transitions. *A*: True non-stationary data-generation process. The structure evolves gradually from the network at the left labeled G_1 to the network at the right labeled G_7. The epochs in which the various networks are active are shown in the horizontal bars, roughly to scale. The horizontal bars represent the segmentation of the 1020 observations, with the transition times labeled below. When these times are known to the algorithm (the KNKT setting), the recovered nsDBN structure is exactly the true structure. *B*: When the times of the transitions are not known (the KNUT setting), the algorithm learns the model-averaged nsDBN structure shown (selecting edges that occur in greater than fifty percent of the sampled structures). The learned networks and most likely transition times are highly accurate (only missing two edges in G_1 and all predicted transition times close to the truth).

true structure is shown in Figure 3A. We chose a small network with features biologically relevant to genetic regulatory networks: a feedback loop, a variable with at least three parents, a pathway of length six, and the inclusion of observed variables that do not even participate in the network.

5.1.1 KNKT SETTING

In this simple setting, the sampler rapidly converges to the correct solution. We generate a data set using the structure in Figure 3A, and run our sampler for 100,000 iterations, with the first 25,000 samples thrown out for burn-in.

To obtain a consensus (model averaged) structure prediction, an edge is considered present at a particular time if the posterior probability of the edge is greater than 0.5. The value of λ_m has no effect in this setting, and the value of λ_s is varied between 0.1 and 50. The predicted structure is exactly identical to the true structure shown in Figure 3A for a broad range of values, $0.5 \leq \lambda_s \leq 10$, indicating robust and accurate learning.

5.1.2 KNUT SETTING

In this setting, transition times are unknown and must be estimated *a posteriori*. The prior on m remains unused, and for the prior on s, the value of λ_s is again varied between 0.1 and 50.

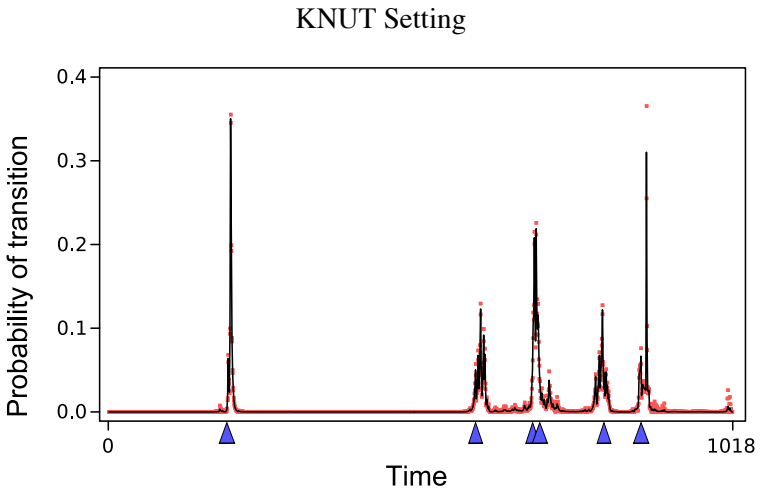


Figure 4: Posterior probability of transition times when learning an nsDBN in the KNUT setting. The blue triangles on the baseline represent the true transition times and the red dots represent one standard deviation from the mean probability, which is drawn as a black line. The variance estimates were obtained using multiple chains. The highly probable transition times correspond closely with the true transition times.

Again, we generate a data set using the structure from Figure 3A and run the sampler for 200,000 iterations, with the first 50,000 samples thrown out for burn-in. More samples are collected in the KNUT setting because we expect that convergence will be slower given the larger space of nsDBNs to explore.

The predicted consensus structure is shown in Figure 3B for $\lambda_s = 5$; this choice of λ_s provides the most accurate predictions. The estimated structure and transition times are very close to the truth. All edges are correct, with the exception of two missing edges in G_1, and the predicted transition times are all within 10 of the true transition times. We can also examine the posterior probabilities of transition times over all sampled structures. This is shown in Figure 4. The blue triangles on the baseline represent the true transition times, and spikes represent transition times that frequently occurred in the sampled structures. While the highest probability regions do occur near the true transition times, some uncertainty exists about the exact locations of t_3 and t_4 since the fourth epoch is exceedingly short.

To obtain more accurate measures of the posterior quantities of interest (such as the locations of transition times), we generate samples from multiple chains; we use 25 in the KNUT setting. Combining the samples from several chains allows us to estimate both the probability of a transition occurring at a certain time and the variance of that estimate. The red dots in Figure 4 represent one standard deviation above and below the estimated mean probability of a transition occurring at a particular time. We discovered that the speed of convergence under the KNKT and KNUT settings were very similar for a given m. This unexpected result implies that the posterior over transition times is rather smooth; therefore, the mixing rate is not greatly affected when sampling transition times.

5.1.3 UNUT SETTING

Finally, we consider the UNUT setting where the number and times of transitions are both unknown. We examine the accuracy of our method in this setting using several values of λ_s and λ_m. We use the range $1 \leq \lambda_s \leq 5$ because we know from the previous settings that the most accurate solutions were obtained using a prior within this range; the range $1 \leq \lambda_m \leq 50$ is selected to provide a wide range of estimates for the prior on m since we have no previous knowledge of what it should be.

Again, we generate a data set using the structure from Figure 3A and run the sampler for 300,000 iterations, with the first 75,000 samples thrown out for burn-in. We collect samples from 25 chains in this setting.

Figure 5 shows the posterior probabilities of transition times for various settings of λ_s and λ_m. As expected, when λ_m increases, the number of peaks decreases. Essentially, when λ_m is large, only the few transition times that *best* characterize the non-stationary behavior of the data will be identified. On the other hand, when λ_m is very small, noises within the data begin to be identified as transition times, leading to poor estimates of transition times.

We can also examine the posterior on the number of epochs, as shown in Figure 6. The largest peak can be used to provide an estimate of m. A smaller λ_m results in more predicted epochs and less confidence about the most probable value of m.

Finally, since we know what the true structure is, we can obtain a precision-recall curve for each value of λ_s and λ_m. The precision-recall curves are shown at the top of Figure 7. To calculate these values, we obtained individual precision and recall estimates for each network at each observation and averaged them over all observations. Therefore, the reported precision and recall values can be viewed as the average precision and average recall over all observations.

One way to identify the best parameter settings for λ_s and λ_m is to examine the best F1-measure (the harmonic mean of the precision and recall) for each. The table in Figure 7 shows the best F1-measures and reveals $\lambda_s = 5$ and $\lambda_m = 1$ as best for this data, although nearly all choices achieve an F1-measure above 0.9.

5.2 Larger Simulated Data Set

To evaluate the scalability of our technique in the most difficult UNUT setting, we also simulate data from a 100 variable network with an average of 50 edges over five epochs spanning 4800 observations, with one to three edges changing between each epoch. We generate 10 different data sets from the model and acquire 25 chains from each data set. For each chain, we take 800,000 samples, with the first 200,000 samples thrown out for burn-in.

The posterior probabilities of transition times and the number of epochs (corresponding to Figures 5 and 6) for one of the simulated data sets are shown in Figure 8. The significantly sharper prediction for the posterior probabilities of transitions occurring at specific times is most likely due to having more observations and, thus, more confident estimates. The number of epochs with the highest posterior probability is five for all choices of priors, which is exactly the true number of epochs for this data set.

Additionally, the precision-recall curves and F1-measures are shown in Figure 9, revealing the $\lambda_s = 1$ and $\lambda_m = 5$ assignments to be best for this data, although all choices achieve excellent results.

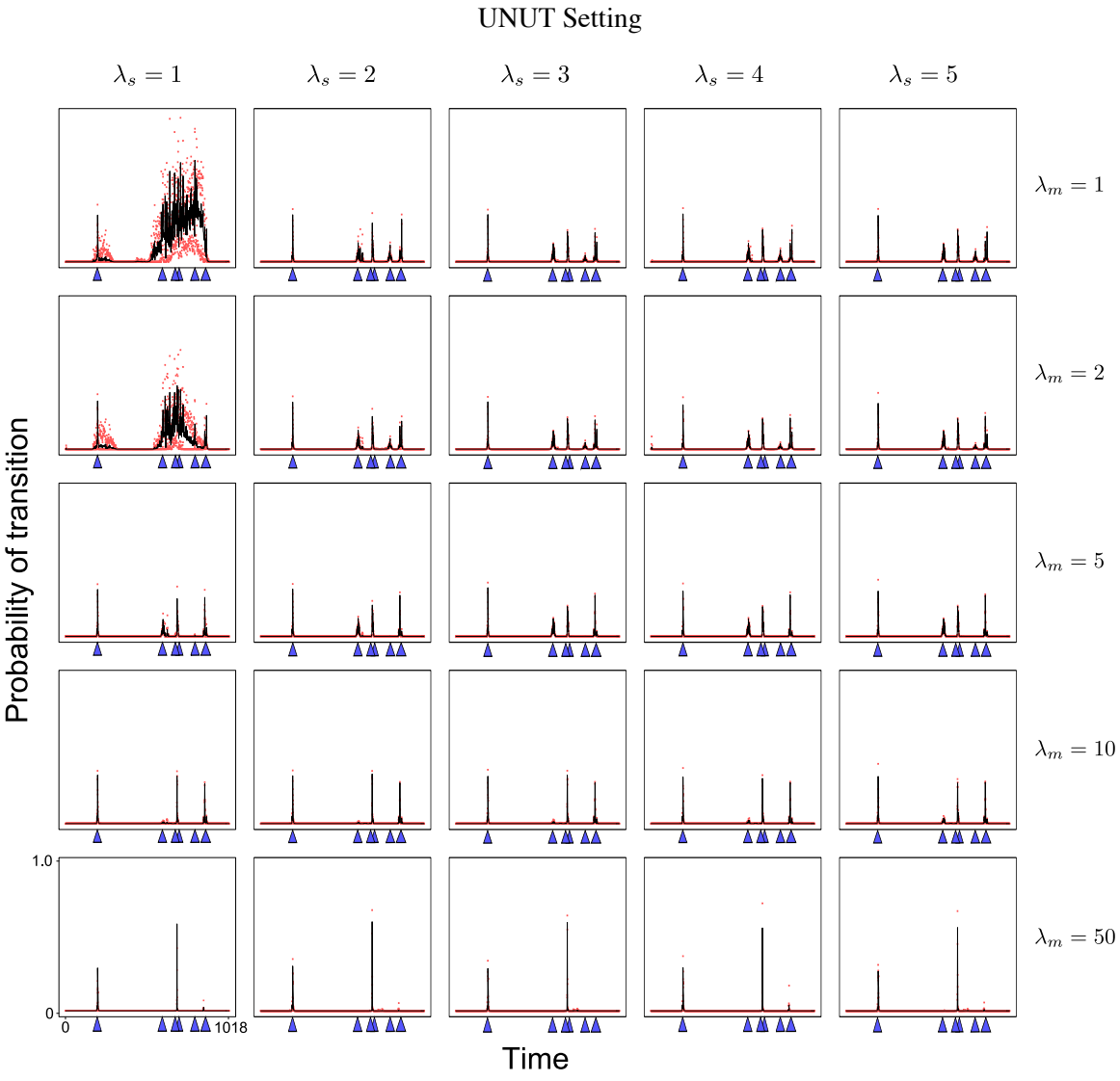


Figure 5: The posterior probabilities of transition times from the sampled structures in the UNUT setting for various values of λ_s and λ_m. As in Figure 4, the blue triangles on the baseline represent the true transition times and the red dots represent one standard deviation from the mean probability obtained from several runs, which is drawn as a black line. Only the (λ_s, λ_m) values of $(1,1)$ and $(1,2)$ seem to result in poor estimates of the true transition times.

5.3 *Drosophila* Muscle Development Gene Regulatory Networks

Having achieved excellent results even in the hardest UNUT setting across ten different data sets generated from two different simulated networks—one with 10 variables and the other with 100—we are confident enough in the usefulness of our model to analyze some real data.

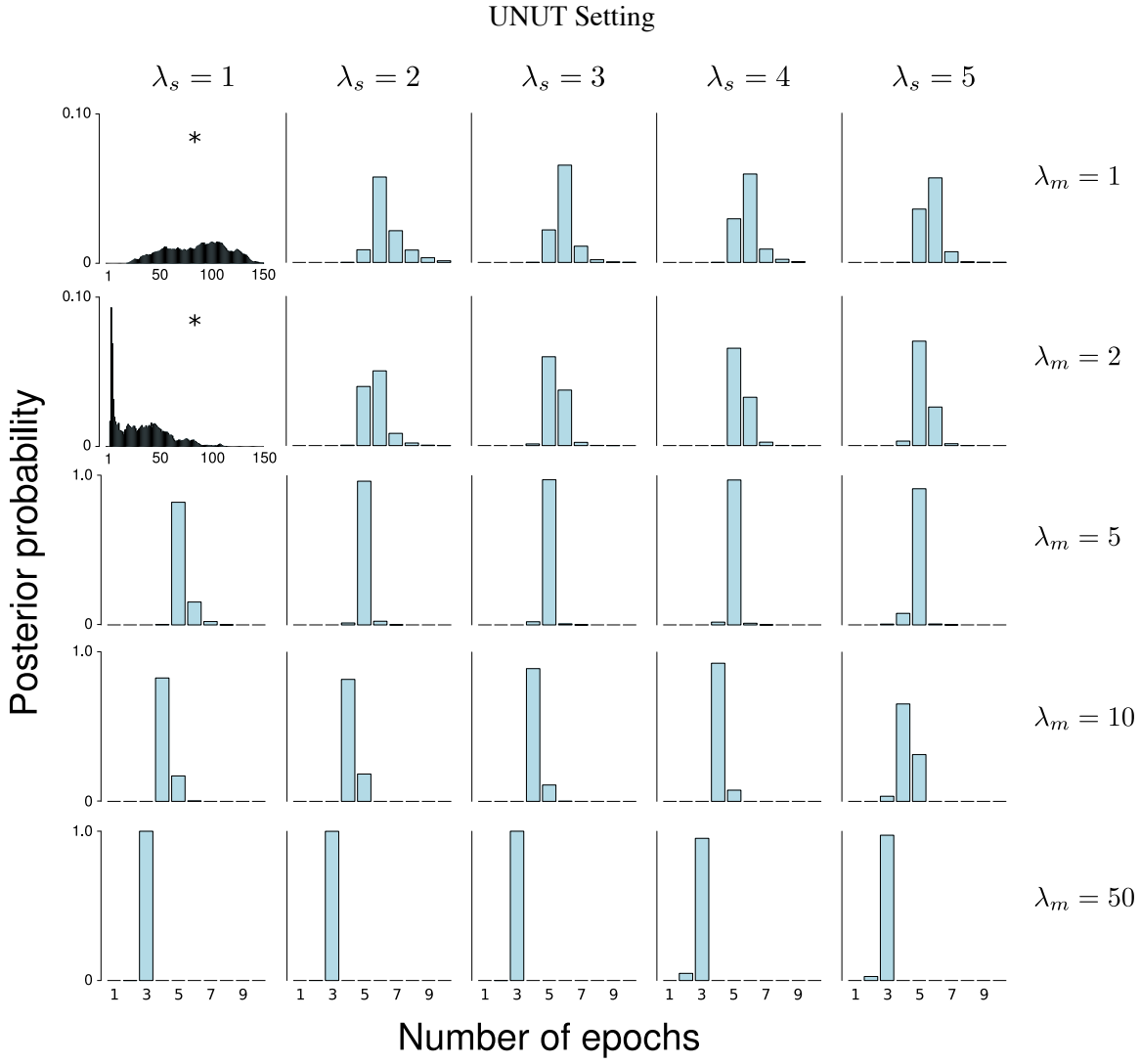
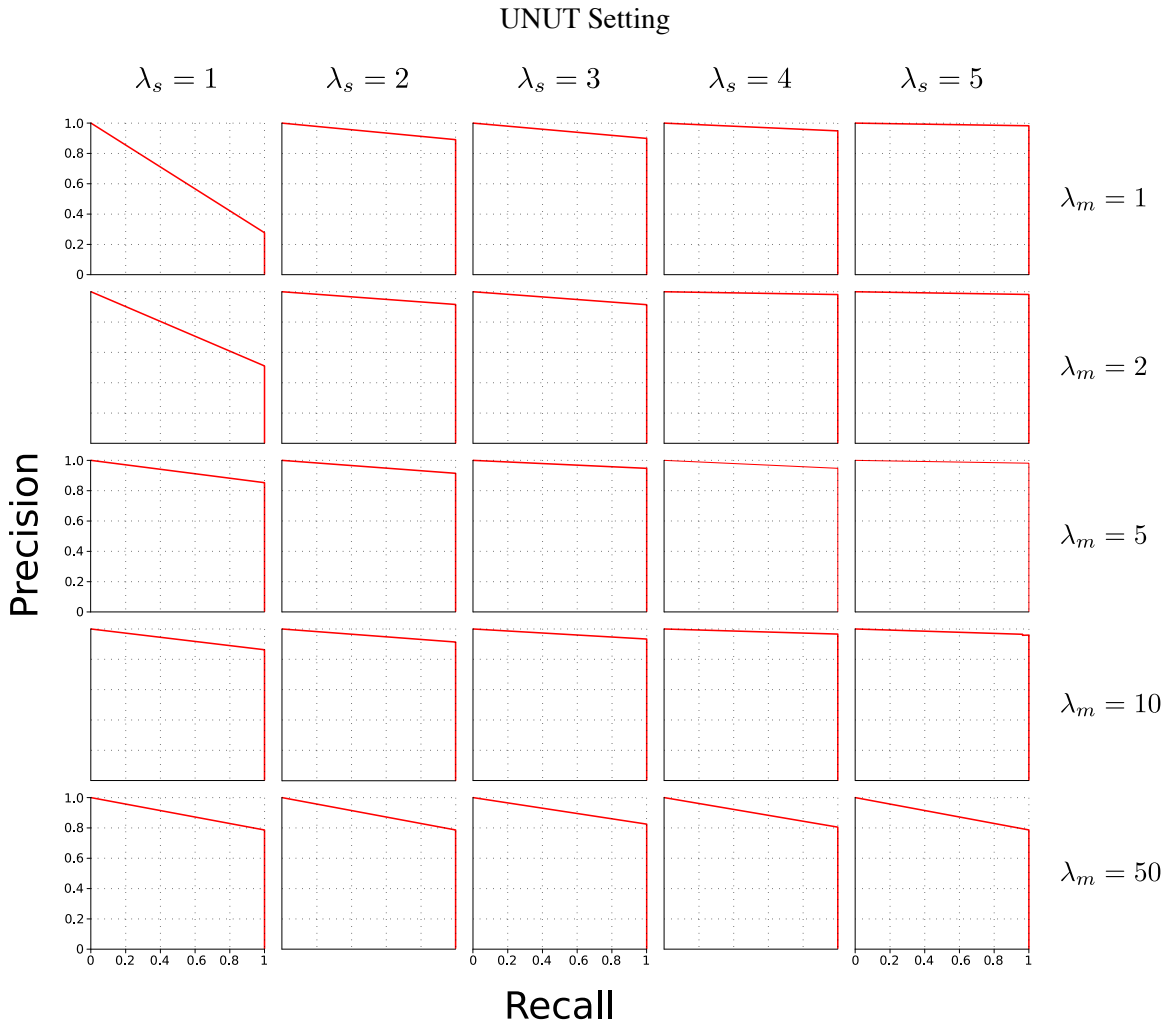


Figure 6: The posterior probabilities of the number of epochs for various values of λ_s and λ_m. The x-axes range from 1 to 10 and the y-axes from 0 to 1 for all of the plots except those marked with a star. The starred plots show predictions with significantly more epochs than the truth. The posterior estimates on the number of epochs *m* are closest to the true value of 7 when λ_m is 1 or 2.

In this subsection, we apply our method to identify non-stationary networks using *Drosophila* development gene expression data from Arbeitman et al. (2002). This data contains expression measurements of 4028 *Drosophila* genes over 66 time steps throughout development and growth during the embryonic, larval, pupal, and adult stages of life. Zhao et al. (2006) focused on 19 genes involved in muscle development and learned a single network over all 66 time steps with this data. Using the same data, Guo et al. (2007) learned a time-varying undirected network over a subset of



λ_s

1	2	3	4	5	
0.4341	0.9423	0.9469	0.9738	**0.9912**	1
0.6760	0.9562	0.9553	0.9906	0.9909	2
0.9206	0.9553	0.9729	0.9731	0.9905	5
0.9264	0.9550	0.9657	0.9829	0.9791	10
0.8804	0.8806	0.9042	0.8922	0.8807	50

λ_m

Figure 7: *Top*: Precision-recall curves for various values of λ_s and λ_m under the UNUT setting. The most accurate estimates for the structure of the nsDBN arise when $\lambda_s = 5$ and $\lambda_m = 1$. *Bottom*: Corresponding F1-measures for the precision-recall curves. F1-measures over 0.9 are shaded; darker shades indicate values closer to 1, and the highest value is shown in bold.

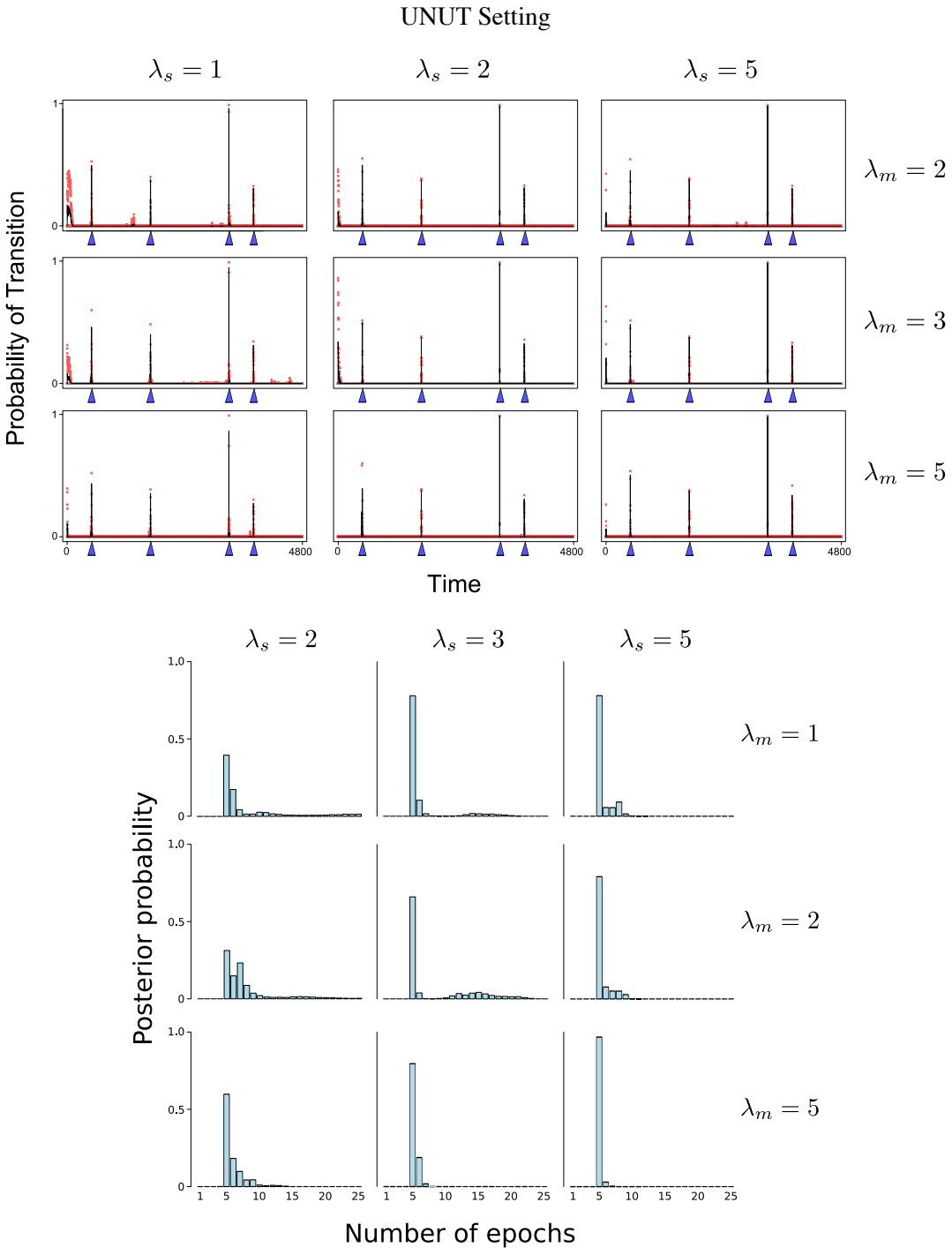
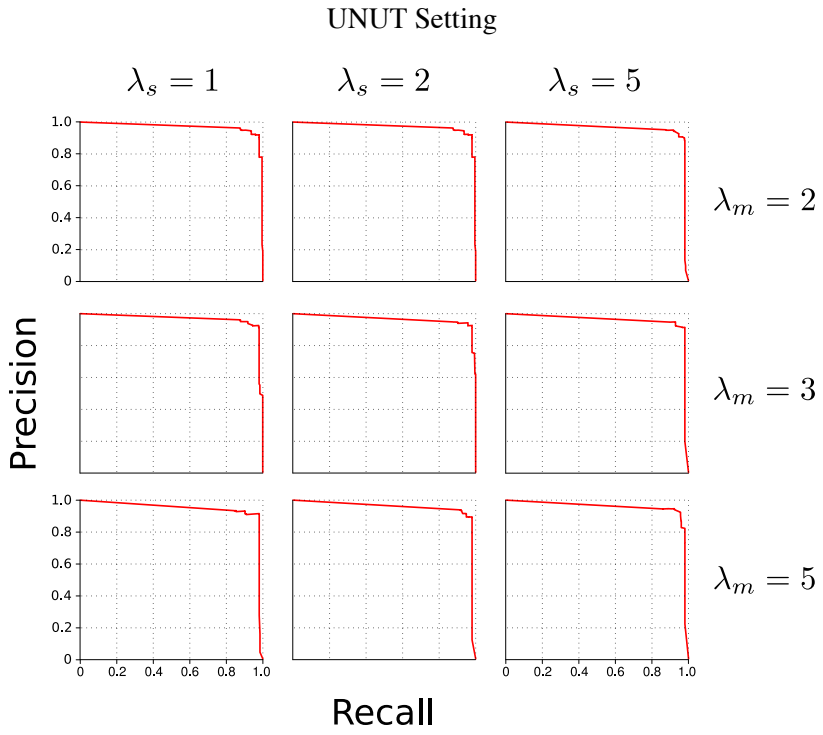


Figure 8: The posterior probabilities of transition times and number of epochs from one of the larger (100 variables, 5 epochs, and 4800 observations) simulated data sets under the UNUT setting for various values of λ_s and λ_m. The axes are the same for all plots. *Top*: The blue triangles on the baseline represent the true transition times and the red dots represent one standard deviation from the mean probability obtained from several runs, which is drawn as a black line.



λ_m \ λ_s	1	3	5
1	0.9489	0.9510	0.9468
2	0.9377	0.9521	0.9356
5	**0.9531**	0.9459	0.9398

Figure 9: *Top*: Precision-recall curves for several values of λ_s and λ_m under the larger 100 variable simulation. *Bottom*: Corresponding F1-measures for the precision-recall curves. F1-measures over 0.9 are shaded; the highest value is shown in bold.

11 of the 19 genes identified by Zhao et al. (2006). To facilitate comparison with as many existing methods as possible, we apply our method to the data describing the expression of the same 11 genes, preprocessing the data in the same way as described by Zhao et al. (2006). Unfortunately, no other techniques predict non-stationary directed networks, so our comparisons are made against the stationary directed network predicted by Zhao et al. (2006) and the non-stationary undirected network predicted by Guo et al. (2007).

We collect 50,000 samples and throw out the first 10,000 for burn-in; we then repeat this process for 25 chains. We need fewer samples in this problem compared to previous data sets because there are relatively few variables and observations.

Figure 10 shows how our predicted structure compares to those reported by Zhao et al. (2006) and Guo et al. (2007). The nsDBN in Figure 10C was learned using the KNKT setting with transition times defined at the borders between the embryonic, larval, pupal, and adult stages.

While all three predictions share many edges, certain similarities between our prediction and one or both of the other two predictions are of special interest. In all three predictions, a cluster seems to form around *myo61f*, *msp-300*, *up*, *mhc*, *prm*, and *mlc1*. All of these genes except *up* are in the myosin family, which contains genes involved in muscle contraction. Within the directed predictions, *msp-300* primarily serves as a hub gene that regulates the other myosin family genes. It is interesting to note that the undirected method predicts connections between *mlc1*, *prm*, and *mhc* while neither directed method makes these predictions. Since *msp-300* seems to serve as a regulator to these genes, the method of Guo et al. (2007) may be unable to distinguish between direct and indirect interactions, due to its undirected nature and reliance on correlations.

Two interesting temporal similarities arise when comparing our predictions to those from Guo et al. (2007). First, an interaction between *eve* and *actn* arise at the beginning of the pupal stage in both methods. Second, the connection between *msp-300* and *up* is lost in the adult network. Note that the loss of this edge actually characterizes the progression to the adult stage from the pupal stage in our prediction, while the method from Guo et al. (2007) combines the two stages. The estimation of a combined pupal/adult stage may simply be due to predicting the loss of the edge between *msp-300* and *up* earlier in development than our method.

Despite the similarities, some notable differences exist between our prediction and the other two predictions. First, we predict interactions from *myo61f* to both *prm* and *up*, neither of which is predicted in the other methods, suggesting a greater role for *myo61f* during muscle development. Also, we do not predict any interactions with *twi*. During muscle development in *Drosophila*, *twi* acts as a regulator of *mef2* which in turn regulates some myosin family genes, including *mlc1* and *mhc* (Sandmann et al., 2006; Elgar et al., 2008); our prediction of no connection to *twi* mirrors this biological behavior. Finally, we note that in our predicted structure, *actn* never connects as a regulator (parent) to any other genes, unlike in the network predicted by Zhao et al. (2006). Since *actn* (actinin) only binds actin, we do not expect it to regulate other muscle development genes, even indirectly.

If we transition to the UNUT setting, we can also examine the posterior probabilities of transition times and epochs. These plots are shown in Figure 11A and 11B, respectively. The transition times with high posterior probabilities correspond well to the embryonic→larval and the larval→pupal transitions, but a posterior peak occurs well before the supposed time of the pupal→adult transition; this reveals that the gene expression program governing the transition to adult morphology is active well before the fly emerges from the pupa, as would clearly be expected. Also, we see that the most probable number of epochs is three to four, mirroring closely the total number of developmental stages.

5.4 Simulated Data Set Similar to the *Drosophila* Data Set

To evaluate the accuracy of a recovered nsDBN on a problem of exactly the same size as the predicted *Drosophila* muscle development network, we simulate a non-stationary time-series with the same number of nodes and a similar level of connectivity as the *Drosophila* data set. We generate data from an nsDBN with 66 observations and transition times at 30, 40, and 58 to mirror the number of observations in embryonic, larval, pupal, and adult stages of the experimental fly data. Since

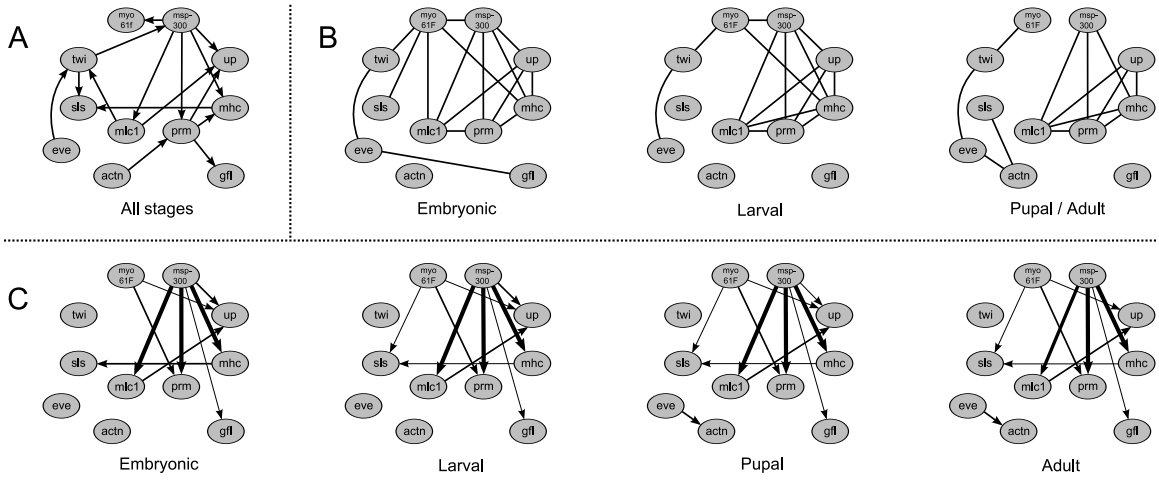


Figure 10: Comparison of computationally predicted *Drosophila* muscle development networks. *A*: The directed network reported by Zhao et al. (2006). *B*: The undirected networks reported by Guo et al. (2007). *C*: The nsDBN structure learned under the KNKT setting with $\lambda_s = 2$. Only the edges that occurred in greater than 50 percent of the samples are shown, with thicker edges representing connections that occurred more frequently.

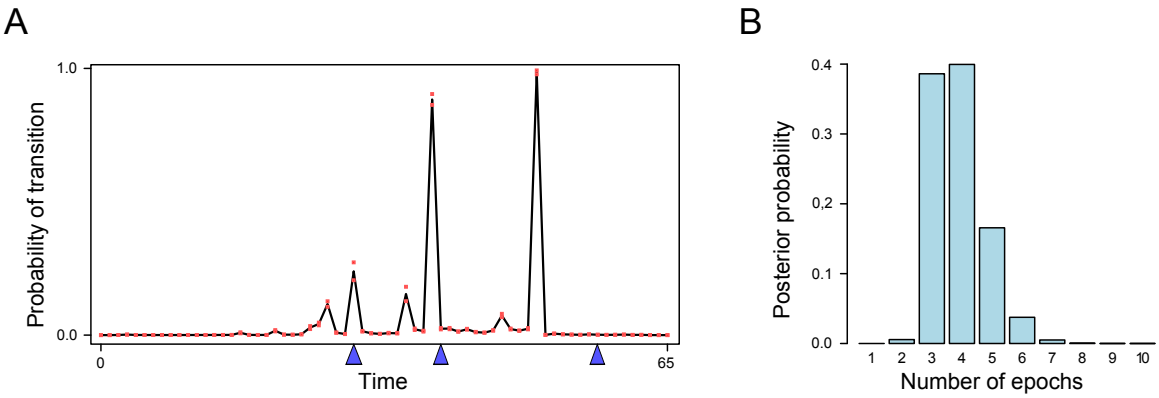


Figure 11: Learning nsDBN structure in the UNUT setting using the *Drosophila* muscle development data. *A*: Posterior probabilities of transition times using $\lambda_m = \lambda_s = 2$. The blue triangles on the baseline represent the borders of embryonic, larval, pupal, and adult stages. *B*: Posterior probability of the number of epochs. The high weight for 3 and 4 epochs closely matches the true number of developmental stages.

it is difficult to estimate the amount of noise in the experimental data, we also simulate noise at various signal-to-noise ratios, from 4:1 down to 1:1. Finally, since many biological processes have more variables than observations, we examine the effect of increasing the number of experimental replicates as a possible means to overcome this challenge.

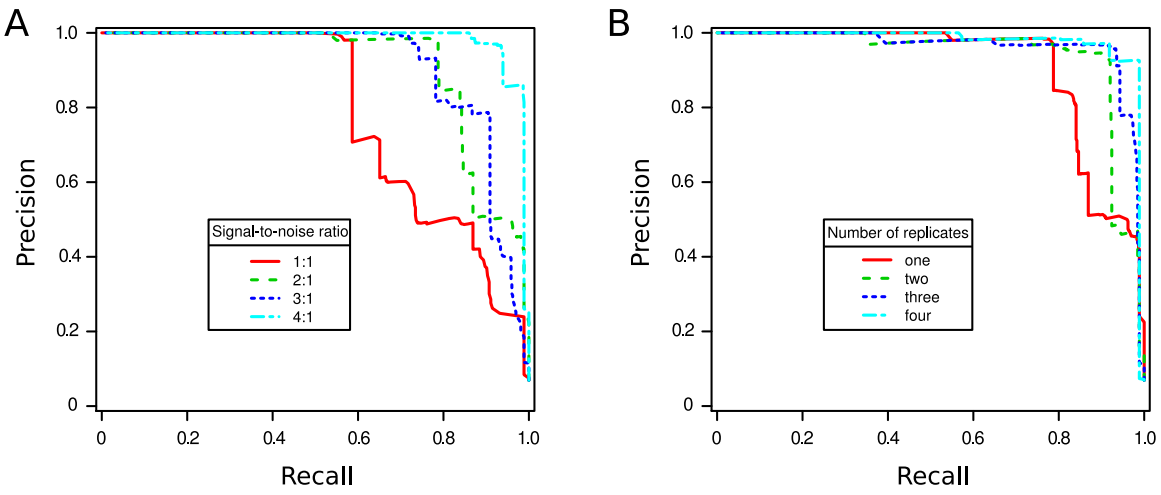


Figure 12: An nsDBN was learned on simulated data that mimicked the number of nodes, connectivity, and transition behavior of the experimental fly data. This allowed us to estimate the accuracy of learned nsDBNs on a problem of this size. *A*: Precision-recall curves for increasing values of the signal to noise ratio in the data (using one replicate). *B*: Precision recall curves for an increasing number of experimental replicates (using an SNR of 2:1). A greater signal to noise ratio and a greater number of experimental replicates both result in better performance, as expected.

As discussed earlier, to obtain posterior estimates of quantities of interest, such as the number of epochs or transition times, we generate many samples from several chains; averaging over chains provides a more efficient exploration of the sample space. To incorporate replicates into the posterior calculations, we generate samples from multiple chains (25) for each set of replicates. Since the underlying data generation process is the same for each replicate, we simply average over all the chains. The results of these simulations are summarized in Figure 12.

As expected, as the signal-to-noise ratio of the data increases, the greater the accuracy in the learned nsDBNs as reflected in the F1-measures: 1:1 is 0.734, 2:1 is 0.850, 3:1 is 0.875, and 4:1 is 0.950. Additionally, increasing the number of replicates also increases prediction accuracy: one is 0.869, two is 0.924, three is 0.945, and four is 0.956. This demonstrates the importance of multiple replicates for biological data with many variables but few observations.

This simulation study allows us to explore how much a relatively small data set and noise affect the predictions from our algorithm on a problem of similar size as the *Drosophila* muscle development network. From the results in Figure 12, we see that learning an nsDBN on a problem of the same scale as the *Drosophila* data set results in accurate network reconstruction, even in the presence of substantial noise. Therefore, we can surmise that any inaccuracies in our predicted *Drosophila* muscle development network would not arise due to the use of a dataset of that size, but might arise from an exceptionally high level of noise in the data (or inappropriate modeling assumptions).

5.5 Neural Information Flow Networks in Songbirds

Our goal is to learn neural information flow networks in the songbird brain. Such networks represent the transmission of information between different regions of the brain. Like roads, the anatomical connectivity of a brain indicates potential pathways along which information can travel. Like traffic, neural information flow networks represent the dynamic utilization of these pathways. By identifying the neural information flow networks in songbirds during auditory stimuli, we hope to understand how sounds are stored and processed in the brain.

In this experiment, eight electrodes were placed into the vocal nuclei of six female zebra finches. Voltage changes were recorded from populations of neurons while the birds were provided with four different two-second auditory stimuli, each presented twenty times. The resulting voltages were post-processed with an RMS transformation and binned to 5 ms; this interval was chosen because it takes 5–10 ms for a neural signal to propagate through one synaptic connection (Kimpo et al., 2003). We analyze data recorded from electrodes for two seconds pre-stimulus, two seconds during stimulus, and two seconds post-stimulus. We learn an nsDBN for two of the birds over six seconds for two different stimuli using all repetitions; this data set contains 8 variables and nearly 25,000 observations for each bird and each stimulus.

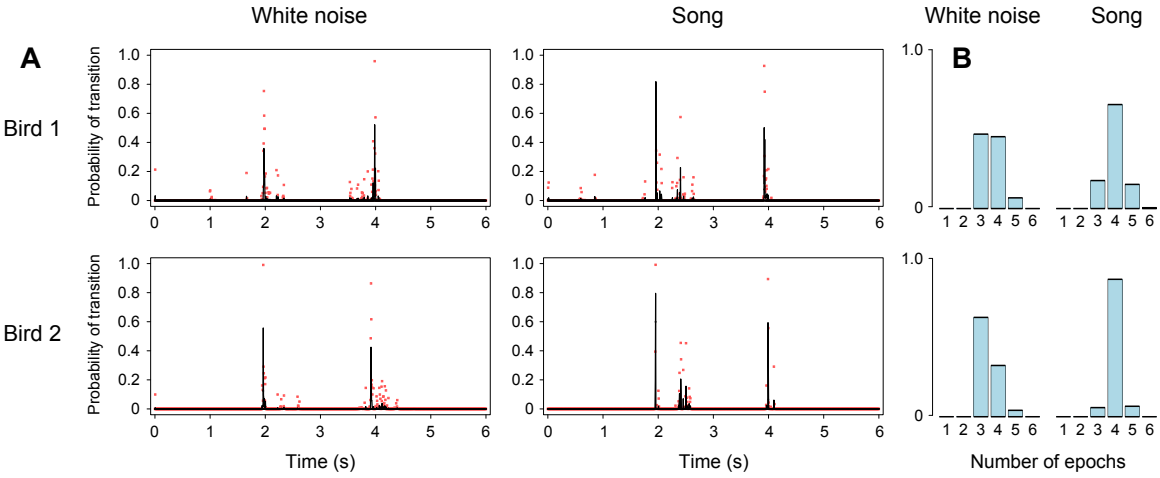


Figure 13: Posterior results of learning nsDBNs under the UNUT setting for two birds presented with two different stimuli (white noise and song). *A*: Posterior transition time probabilities. Transitions are consistently predicted near the stimulus onset (2 seconds) and offset (4 seconds). *B*: Posterior estimates of the number of epochs. The estimated number of epochs is three or four, with strong support for the value four when the bird is presented with a song.

The posterior transition time probabilities and the posterior number of epochs for two birds presented with two different stimuli under the UNUT setting ($\lambda_s = \lambda_m = 2$) can be seen in Figure 13. Note how the estimated transition times correspond closely with the times of the stimulus onset and offset and how the posterior estimate of the number of epochs is around three or four; taken together, these statistics imply that different networks predominate in the pre-stimulus, during stimulus, and post-stimulus time periods.

The posterior estimates consistently differ when the bird is listening to white noise versus song. When listening to a song, an additional transition is predicted 300–400 ms after the onset of a song but not after the onset of white noise. This implies that the bird further analyzes a sound after recognizing it (e.g., hearing a known song), but performs no further analysis when it does not recognize a sound (e.g., hearing white noise).

Previous analysis of this data assumed that any changes in the neural information flow network of a songbird listening to sound occurred only at sound onset and offset (Smith et al., 2006). Only by appropriately modeling the neural information flow networks as nsDBNs are we able to learn that this assumption is not accurate.

Further analysis and investigation of this data is left to future work.

5.6 Performance and Scalability

Due to the use of efficient data structures in the sampler implementation, the computational time needed to update the likelihood is essentially the same for all moves. Therefore, the runtimes of the algorithm under the KNKT, KNUT, and UNUT settings do not differ for a given number of samples. Nevertheless, one typically wants more samples in settings with increased uncertainty to ensure proper convergence.

For the small ten variable simulated data set, the sample collection process for each chain takes about 10 seconds per 100,000 samples, which translates to 10 seconds for the KNKT setting, 20 seconds for the KNUT setting, and 30 seconds for the UNUT setting. Fortunately, all runs can easily be executed in parallel. Sample collection for the *Drosophila* data set and the similarly sized simulated data set takes only a few seconds for each chain under all settings.

For the larger 100 variable simulated data set, sample collection takes about 2 minutes per 100,000 samples. The increased runtime is primarily due to the larger number of variables, so defining the neighborhood for each move takes more time. Due to intelligent caching schemes, the number of observations affects runtime in only a sublinear fashion (provided that enough memory is available).

Surprisingly, one of the largest contributors to running time is the actual recording of the MCMC samples. For example, each sample in the larger simulated data set can be represented by a 10,000 by 4,800 binary matrix of indicators for individual edges at every point in time. A full recording of each sample is therefore very time consuming: just recording each sample in the larger simulated data set leads to an increased runtime of about 50 minutes per 100,000 samples. We can alleviate this problem in several ways. First, because only a small number of those 48 million values actually change between samples, each sample can be represented and output in a compressed fashion; however, the same amount of processing still must occur after the sample collection completes. A better option is to only record the posterior quantities of interest. For example, recording just the transition times and number of epochs adds only a few seconds to the runtime on the larger simulated data set.

6. Discussion

Non-stationary dynamic Bayesian networks provide a useful framework for learning Bayesian networks when the generating processes are non-stationary. Using the move sets described in this paper, nsDBN learning is efficient even for networks of 100 variables, is generalizable to situations of varying uncertainty (KNKT, KNUT, and UNUT), and is robust (i.e., not very sensitive) to the

choice of hyper-parameters over a large range of values. Additionally, by using a sampling-based approach, our method allows us to assess a confidence for each predicted edge—an advantage that neither Zhao et al. (2006) nor Guo et al. (2007) share.

We have demonstrated the feasibility of learning an nsDBN in all three settings using simulated data sets of various numbers of transition times, observations, variables, epochs, and connection densities. Additionally, we have identified nsDBNs in the KNKT and UNUT settings using biological gene expression data. The *Drosophila* muscle development network we predict is consistent with the predictions from other techniques and conforms to many known biological interactions from the literature. The predicted transition times and number of epochs also correspond to the known times of large developmental changes. Although each connection on the predicted *Drosophila* muscle development network is difficult to verify, simulated experiments of a similar scale demonstrate highly accurate predictions, even with moderately noisy data and one replicate.

While we focus on certain aspects of the model in this paper, many of our decisions are choices rather than restrictions. For example, we present results using a Markov lag of one, but any Markov lag could be used. Additionally, we use the BDe score metric, but any score metric or conditional independence test can be used instead; however, any score metric which does not integrate over the non-structural parameters would require an augmented sampling procedure. The assumption of discrete data is not necessary; our method easily extends to continuous data, provided that an appropriate scoring metric (like BG) is adapted.

A discrete view of time is necessary to our approach, but many continuous-time data sets can be transformed into discrete-time ones without significant loss of information. The use of directed graphs is also necessary, and desired, but undirected estimates can be obtained through moralization of directed estimates. Although we choose simple priors to learn smoothly evolving networks, nearly any priors would be easy to incorporate; in particular, incorporating expert knowledge about the problem domain would be an ideal method for defining priors.

For problems of more than a few variables, the use of MCMC sampling is essential since EM techniques would not converge in any reasonable time frame given such a large sample space. One of our key discoveries for increasing convergence is the reformulation of the problem from learning multiple networks to learning a network and changes to that network. This parameterization provides an intuitive means for defining evolving networks and allows us to define move sets with good convergence properties. Our particular choices of the move sets are not the only possible ones, but we have taken extra care to ensure that they work well on the types of problems we examine in this paper.

The proposed sampling algorithm scales well to problems with hundreds of variables and thousands of observations, but we are not certain how well it will scale to problems that are orders of magnitude larger. When obtaining sample runs takes days instead of minutes or hours, it may become desirable to obtain faster estimates, even if they are approximate. Variational methods are one alternative to MCMC sampling approaches, often obtaining faster estimates at the cost of decreased accuracy (Beal and Ghahramani, 2006). For the scale of problems in this paper, the run times and convergence rates of our MCMC sampling algorithm were good enough that we did not have to resort to variational approximations. However, if we wish to explore much larger data sets in the future, we may need to develop a variational algorithm to obtain results in a reasonable time frame.

Non-stationary DBNs offer all of the advantages of DBNs (identifying directed, potentially non-linear interactions between variables in multivariate time-series) and are additionally able to identify non-stationarities in the interactions between variables. The sampling algorithm presented

here allows one to estimate the strength of individual edges as well as posterior distributions of and quantities of interest. In future work, we hope to analyze data from other fields that have traditionally used DBNs and instead use nsDBNs to identify and model previously unknown or uncharacterized non-stationary behavior.

Another direction that could be explored in the future is nsDBN learning with latent variables. Following an EM approach like in Friedman (1997) would be the first step, but one would also need to consider how to connect hidden variables across epochs and how to incorporate different numbers of hidden variables at different epochs. Learning non-stationary networks with latent variables seems to present a vexing challenge in the general case, but might be feasible in simple cases when enough data is available.

Acknowledgments

The authors would like to thank the reviewers for their insightful comments and suggestions. A.J.H. gratefully acknowledges funding for this work from an NSF CAREER award (0347801), an Alfred P. Sloan Fellowship, and a grant from NIH in Collaborative Research in Computational Neuroscience (R01-DC007996-01).

Appendix A.

While we decided to place a (truncated) geometric prior on the number of epochs m, other priors may be considered. Talih and Hengartner (2005) chose to model epoch lengths as i.i.d. geometric random variables. Here, we prove that a geometrically distributed prior on epoch lengths is equivalent to a geometrically distributed prior on the number of epochs.

Letting l_i be the length of epoch i and N be the total number of observations, we can write a geometrically distributed prior on epoch lengths as:

$$\begin{aligned}\prod_{i=1}^{m}(1-p)^{l_i-1}p &= p^m\prod_{i=1}^{m}(1-p)^{l_i-1} \\ &= p^m(1-p)^{\sum_{i=1}^{m}l_i-1} \\ &= p^m(1-p)^{N-m} \\ &= \left(\frac{p}{1-p}\right)^m(1-p)^N.\end{aligned}$$

Since N is the same for every nsDBN, we see that the geometrically distributed prior is simply a function of the number of epochs m. Compare this to a geometrically distributed prior on the number of epochs, shown below:

$$\begin{aligned}(1-q)^{m-1}q &= (1-q)^m\frac{q}{1-q} \\ &= A(1-q)^m.\end{aligned}$$

where A is a constant that is the same for all nsDBNs. Therefore, a geometrically distributed prior on epoch lengths with success probability $p=\frac{1-q}{2-q}<1/2$ is equivalent to a geometrically distributed prior on the number of epochs with success probability q. In this paper, we assume that q takes the form $q=1-\mathrm{e}^{-\lambda}$, allowing for a more intuitive control of the prior by simply changing λ.

References

Amr Ahmed and Eric P. Xing. TESLA: Recovering time-varying networks of dependencies in social and biological studies. *Proceedings of the National Academy of Sciences*, 106(29):11878–11883, 2009.

Michelle N. Arbeitman, Eileen E.M. Furlong, Farhad Imam, Eric Johnson, Brian H. Null, Bruce S. Baker, Mark A. Krasnow, Matthew P. Scott, Ronald W. Davis, and Kevin P. White. Gene expression during the life cycle of *Drosophila melanogaster*. *Science*, 5590(297):2270–2275, Sep 2002.

Matthew J. Beal and Zoubin Ghahramani. Variational Bayesian learning of directed graphical models with hidden variables. *Bayesian Analysis*, 1(4):793–832, 2006.

Allister Bernard and Alexander J. Hartemink. Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. In *Pacific Symposium on Biocomputing*, volume 10, pages 459–470. World Scientific, Jan 2005.

Wray Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, 1996.

Carlos M. Carvalho and Mike West. Dynamic matrix-variate graphical models. *Bayesian Analysis*, 2(1):69–98, 2007.

David Maxwell Chickering, Dan Geiger, and David Heckerman. Learning Bayesian networks is NP-Hard. Microsoft Research Technical Report MSR-TR-94-17, Microsoft, Nov 1994.

David Maxwell Chickering, Dan Geiger, and David Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*, pages 112–128. Society for Artificial Intelligence in Statistics, Jan 1995.

Lonnie Chrisman. A roadmap to research on Bayesian networks and other decomposable probabilistic models. CMU technical report, School of Computer Science, CMU, May 1998.

Luís Miguel de Campos, Juan M. Fernandez-Luna, José Antonio Gámez, and José Miguel Puerta. Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, 31(3):291–311, Nov 2002.

Stuart J. Elgar, Jun Han, and Michael V. Taylor. *mef2* activity levels differentially affect gene expression during *Drosophila* muscle development. *Proceedings of the National Academy of Sciences*, 105(3):918–923, Jan 2008.

Nir Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the 14th International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann Publishers, 1997.

Nir Friedman and Zohar Yakhini. On the sample complexity of learning Bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 274–282. Morgan Kaufmann Publishers Inc., Oct 1996.

Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI98)*, pages 139–147. Morgan Kaufmann Publishers Inc., 1998.

Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian networks to analyze expression data. In *Research in Computational Molecular Biology (RECOMB00)*, volume 4, pages 127–135. ACM Press, Apr 2000.

Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996, Apr 2000.

Paolo Giudici and Robert Castelo. Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 50(1–2), Jan 2003.

Paolo Giudici, Peter Green, and Claudia Tarantola. Efficient model determination for discrete graphical models. Technical report, Athens University of Economics and Business, 1999.

Marco Grzegorczyk, Dirk Husmeier, Kieron D. Edwards, Peter Ghazal, and Andrew J. Millar. Modelling non-stationary gene regulatory processes with a non-homogeneous Bayesian network and the allocation sampler. *Bioinformatics*, 24(18):2071–2078, Jul 2008.

Fan Guo, Wenjie Fu, Yanxin Shi, and Eric P. Xing. Reverse engineering temporally rewiring gene networks. In *NIPS workshop on New Problems and Methods in Computational Biology*, Dec 2006.

Fan Guo, Steve Hanneke, Wenjie Fu, and Eric P. Xing. Recovering temporally rewiring networks: A model-based approach. In *Proceedings of the 24th International Conference on Machine Learning (ICML07)*, Jun 2007.

Steve Hanneke and Eric P. Xing. Discrete temporal models of social networks. In *Workshop on Statistical Network Analysis at the 23rd International Conference on Machine Learning*, Jun 2006.

Alexander J. Hartemink, David K. Gifford, Tommi S. Jaakkola, and Richard A. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Pacific Symposium on Biocomputing*, volume 6, pages 422–433. World Scientific, Jan 2001.

David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, Sep 1995.

Reimar Hofmann and Volker Tresp. Discovering structure in continuous variables using Bayesian networks. In *Advances in Neural Information Processing Systems 8 (NIPS95)*, pages 500–506. MIT Press, Dec 1995.

Rhea R. Kimpo, Frederic E. Theunissen, and Allison J. Doupe. Propagation of correlated activity through multiple stages of a neural circuit. *Journal of Neuroscience*, 23(13):5750–5761, 2003.

Mladen Kolar, Le Song, Amr Ahmed, and Eric P. Xing. Estimating time-varying networks. *The Annals of Applied Statistics*, 4(1):94–123, Mar 2010.

Paul K. Krause. Learning probabilistic networks. *The Knowledge Engineering Review*, 13(4):321–351, 1998.

Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10(4):269–293, Jul 1994.

Pedro Larrañaga, Mikel Poza, Yosu Yurramendi, Roberto H. Murga, and Cindy M.H. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Journal on Pattern Analysis and Machine Intelligence*, 18(9):912–926, Sep 1996.

Nicholas M. Luscombe, M. Madan Babu, Haiyuan Yu, Michael Snyder, Sarah A. Teichmann, and Mark Gerstein. Genomic analysis of regulatory network dynamics reveals large topological changes. *Nature*, 431:308–312, Sep 2004.

David Madigan, Jeremy York, and Denis Allard. Bayesian graphical models for discrete data. *International Statistical Review*, 63(2):215–232, Aug 1995.

Dimitris Margaritis. Distribution-free learning of Bayesian network structure in continuous domains. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI05)*, pages 825–830. AAAI Press / The MIT Press, Jul 2005.

Kevin Murphy. Learning Bayesian network structure from sparse data sets. UC Berkeley technical report 990, Computer Science Department, University of California at Berkeley, May 2001.

Thomas Sandmann, Lars J. Jensen, Janus S. Jakobsen, Michal M. Karzynski, Michael P. Eichenlaub, Peer Bork, and Eileen E.M. Furlong. A temporal map of transcription factor activity: *mef2* directly regulates target genes at all stages of muscle development. *Developmental Cell*, 10(6): 797–807, Jun 2006.

V. Anne Smith, Erich D. Jarvis, and Alexander J. Hartemink. Influence of network topology and data collection on network inference. In *Pacific Symposium on Biocomputing*, volume 8, pages 164–175. World Scientific, Jan 2003.

V. Anne Smith, Jing Yu, Tom V. Smulders, Alexander J. Hartemink, and Erich D. Jarvis. Computational inference of neural information flow networks. *PLoS Computational Biology*, 2(11): 1436–1449, Nov 2006.

Joe Suzuki. Learning Bayesian belief networks based on the minimum description length principle: An efficient algorithm using the branch and bound technique. In *Proceedings of the 13th International Conference on Machine Learning (ICML96)*, pages 462–470. Morgan Kaufmann Publishers Inc., Jul 1996.

Makram Talih and Nicolas Hengartner. Structural learning with time-varying components: Tracking the cross-section of financial time series. *Journal of the Royal Statistical Society B*, 67(3):321–341, Jun 2005.

Claudia Tarantola. MCMC model determination for discrete graphical models. *Statistical Modelling*, 4(1):39–61, Apr 2004.

Stanley Wasserman and Philippa E. Pattison. Logit models and logistic regressions for social networks: I. An introduction to Markov graphs and $p*$. *Psychometrika*, 61(3):401–425, Sep 1996.

Xiang Xuan and Kevin Murphy. Modeling changing dependency structure in multivariate time series. In *Proceedings of the 24th International Conference on Machine Learning (ICML07)*, Jun 2007.

Wentao Zhao, Erchin Serpedin, and Edward R. Dougherty. Inferring gene regulatory networks from time series data using the minimum description length principle. *Bioinformatics*, 22(17): 2129–2135, Sep 2006.